USING IMPRINT TO GUIDE EXPERIMENTAL DESIGN OF SIMULATED TASK ENVIRONMENTS

THESIS

Gregory W. Dye, Civ, USAF

AFIT-ENG-MS-15-J-052

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-MS-15-J-052

USING IMPRINT TO GUIDE EXPERIMENTAL DESIGN WITH SIMULATED TASK ENVIRONMENTS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Gregory W. Dye, BS

Civilian, USAF

June 2015

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-MS-15-J-052

USING IMPRINT TO GUIDE EXPERIMENTAL DESIGN WITH SIMULATED TASK ENVIRONMENTS

Gregory W. Dye, BS

Civ, USAF

Committee Membership:

Dr. Brett J. Borghetti
Chair

Maj. Christina F. Rusnock
Member

Dr. Timothy H. Lacey
Member

AFIT-ENG-MS-15-J-052

Abstract

Experimental designs involving Simulated Task Environments aim to explore interesting conditions with human subjects. By using activity simulators such as IMPRINT, it may be possible to identify these conditions of interest without the need for human subjects. This thesis presents research that aims to demonstrate that IMPRINT can be used to predict human performance in a task environment representing the task performed by Network Analysts of the 33rd Network Warfare Squadron. The research is done by examining the task performed by the Network Analysts, and then designing a Simulated Task Environment modeled on this task. A model of the task performed is also built in IMPRINT. With a first iteration, it was found that the IMPRINT model was not able to predict performance in a majority of cases, however the methodology illustrates a starting point that others may use.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Brett Borghetti, for his guidance and support throughout the course of this thesis effort. The insight and experience was certainly appreciated. In addition, I would like to acknowledge Dr. Tim Lacey and Maj. Christina Rusnock for being on my committee and for their guidance throughout this research. I would, also, like to thank my sponsors at the Human Performance Wing of Air Force Research Labs, for both the guidance and latitude provided to me in this endeavor. In addition, I would like to thank my friends who took the time to help me proofread and edit various sections of this thesis. I would also like to thank James Okolica for providing programming support. Lastly, I would like to thank the subject matter experts who took the time to provide the information that became the foundation for my research.

Gregory W. Dye

Table of Contents

List of Figures

List of Tables

Page

USING IMPRINT TO GUIDE EXPERIMENTAL DESIGN OF SIMULATED TASK ENVIRONMENTS

I. Introduction

Section 1.1: General Issue

Designing effective experiments is one of the keys to successful research. One aspect of experimental design is ensuring that the conditions chosen to explore represent enough of the domain of interest to successfully answer researcher questions, something that is important to Human Factors researchers, those who seek to learn how the human mind and body responds to external stimulus. Within the Air Force, the 711th Human Performance Wing of the Air Force Research Labs (711th, AFRL) performs research on the jobs performed by warfighters. They attempt to discover how task demands affect warfighters, and what tools can be provided to allow the warfighters to better perform their jobs. This research is often done in lab environments, where Simulated Task Environments (STE) are built to replicate the look and feel of the task, along with the decisions required to be made, but simplified to a degree that those not familiar with the task can be used as test subjects. By expanding the set of possible subjects to those who are not familiar with the task, it becomes possible to see how a person would react under the stressful situations that warfighters may find themselves in, without needing to pull the warfighters away from their duties.

Researchers need to explore a variety of conditions in their experiment, and to do so, they need to identify what settings can use. The insight researchers can gain from their experiments can lead to techniques to improve the performance of real warfighters.

These improvements can come in the form of better interfaces, adaptive automation, or better training tools. However, the experiments are inherently limited by the conditions that they explore. If Human Factors researchers want to explore what happens during the most difficult situations, then their conditions must push the subject to the degree that they are overwhelmed, otherwise the research question(s) cannot be answered fully. Likewise, if Human Factors researchers want to examine situations where the task demand is so light the subject may become underloaded to the degree that they become bored, but the conditions are all engaging, then the experiment doesn't provide adequate results. If all conditions represent extreme conditions, then the experiment may not have any baseline conditions, where it is possible to see how the subject reacts and performs in a normal condition that can be compared with the more extreme situations.

It can be difficult to know what conditions will put the subject in the desired levels of workload without running an experiment with human subjects, but this process can be expensive. A small pilot study with a limited number of subjects may allow the conditions of the experiment to be tuned before a larger, formal study is performed. Although a pilot study may be less resource intensive than a full study, it can still be time-consuming and costly. If these pilot studies could be augmented with simulations, then it would dramatically cut down on the time and resource constraints, and provide additional data that could be used to guide experimental design.

One of the areas of interest to 711th HPW is studying those who perform tasks in the cyber domain. The United States Air Force's computer networks are under almost constant attack (Champion, Rajivan, Cooke, & Janwala, 2012). Some of these attacks are as simple as phishing emails, while others are complicated hacking attempts with the goal to extract valuable data from the Air Force’s computers. One of the first human lines of defense against these attacks is the Network Analysts of the 33rd Network Warfare Squadron (NWS). These analysts are tasked with the job of watching the edge of the Air Force network, and monitoring for suspicious activity. To facilitate this task, the analysts use ArcSight, a commercial software tool, to compile data from numerous sensors including email servers to intrusion detection systems (Lovell, 2014; Rice, 2014). ArcSight displays alerts which represent all the possibly suspicious activity. However, even with all this information in one place, a single analyst must respond to hundreds of different alerts in a single eight hour shift, attempting to find the true threats among the many alerts (Lovell, 2014).

Performing this task without any mistakes is very difficult (Liu, Erbacher, Glodek, Etoty, & Yen, 2013). Some events that are not threats will be sent for investigation, wasting government personnel time and resources, and in other cases, a real threat will go undetected which can compromise the Air Force’s valuable information (D'Amico & Whitley, 2007). Many times these mistakes are caused by human error. Some of these errors may be caused by an analyst being over-tasked, where they must either rush through the alerts, being more likely to make mistakes, or simply ignore the alerts they don’t have time for, letting any of these alerts that represent true threats go on undetected. The task is further complicated since most sensors are set at a threshold to generate many false positives in order to avoid any false negatives. Many of these thresholds are set in such a way that only 1% of alerts are identified by the analysts as representing true threats, all the rest being false alerts. For some categories of alerts, the threshold is orders of magnitude worse (Lovell, 2014). If it were possible to know when an analyst was being over-worked, then it might be possible for adjustments to be made to minimize these mistakes. These adjustments could include assigning additional staff, or adjusting the filters on the alerts to reduce the amount of alerts a single analyst must investigate.

Section 1.2: Problem Statement

Human Factors experiments can rely on possibly unfounded assumptions about how various factors may affect performance and workload. If these factors are improperly tuned, then the experiment may not span the breadth of interesting workloads.

In addition, currently there is no simulated task environment dedicated to replicating the workload experienced by Air Force network analysts. Thus, it is difficult to fully understand the analysts' mental state as they monitor the network.

Section 1.3: Research Objectives/Questions/Hypotheses

The first objective of this research is to investigate how workload analyzers such as the Improved Performance Research Integration Tool (IMPRINT) (Army Research Laboratory, 2010) can be used to guide experimental design by modeling how manipulation of factors would affect workload and performance, and then to demonstrate that this model can be utilized to develop an experimental design for a human-in-the-loop study.

The second objective of this research is to present the steps required to build a simulated task environment that emulates the cognitive demand that network analysts encounter. A sub-task of this objective includes performing a cognitive task analysis on the task performed by network analysts. The cognitive task analysis includes examining the decisions they are required to make, the thought process they use, the information they must process to make the decisions, and the environment that they work in. Once the cognitive task analysis is complete, it is possible to use this knowledge to construct the simulated task environment, and the IMPRINT model

Section 1.4: Methodology

The first step of this research is to identify a specific task to explore, in this case, the task performed by the Line Analysts of the 33rd Network Warfare Squadron (33rd NWS), whose duties include monitoring the network traffic at the edge of the Air Force's networks. The Line Analysts have the first look at the data and filter the alerts that need to be explored at a deeper level. The details of the task are explored by performing a cognitive task analysis. This cognitive task analysis is first done by performing a literature review, and then by interviewing those with experience in the field, either working as analysts, or working closely with analysts. With the information from the interviews and literature, a workflow diagram is developed to model the decisions analysts make and the flow of information (Crandall, Klein, & Hoffman, 2006). With the information from the workflow diagram, the simulated task environment can be designed. The design process is very iterative, working with programmers to outline everything the task environment needs to do, and considering the tradeoffs that need to be made. These tradeoffs include considering the time and memory constraints that would be imposed by the amount of data being logged, or the amount of time that implementing a feature may take. The design process also requires further research about the task, as details that had not been considered before may need to be explored to properly design the STE.

In conjunction with the simulated task environment's design, the task is modeled in IMPRINT, a workload modeling tool that makes it possible to simulate how changing factors would affect both the operator's workload and performance (Army Research Laboratory, 2010). IMPRINT makes it possible to identify which combinations of factors will drive workload and performance to the desired amounts, which can then be used in a Calibration Study using the Simulated Task Environment.

Section 1.5: Assumptions/Limitations

There are three different categories of assumptions and limitations with this research. The first category is assumptions regarding the conceptual model of the task. The second category is assumptions regarding the STE, and the STE experiments. The third category is assumptions regarding the IMPRINT model.

A limitation of the conceptual model is caused by this research focusing only on the task performed by the 33rd NWS, and as such, its applicability may be limited for other network defense environments. The research also focuses only on Line Analysts, those who perform the first level analysis on the alerts that enter the system, so the research is limited beyond this scope.

A limitation regarding the STE design is that the task environment only emulates the single task of identifying malicious activity from network-generated alerts. As the analysts do not do anything to respond to the incident beyond creating a report and sending it to the incident response team, this task environment is also limited in this scope. The task environment is constructed for use by a single person at a time, which is a contrast to the environment of a network analyst where a small team work closely together. Communication can be simulated to indicate incoming messages to the operator, though simulating the communication does not offer the same effect as with real people. The experiments performed with the STE are also limited by time, so the subjects will not experience the same level of fatigue during a 10 minute trial that a real analyst would over an 8 hour shift. The real world task may include interruptions, such as a message from a supervisor that could pull them away from the task, while the subjects using the STE are able to focus on the task for the full 10 minutes. There is also the assumption that the simplification done to model the task with the STE won't change the required cognitive thought process.

Further, this research is limited by the natural difference between subject matter experts, and the novices typically used for Human Factors experiments. Due to this difference, it cannot be guaranteed that the two would make the same decisions. This difference is caused by different degrees of background knowledge about the task. A further difference is the habits and techniques an expert will have developed over years of experience as opposed to someone with only limited training and knowledge of the task. One of the differences between the two is how much trust the user gives the system. An expert may know that the system should be trustworthy in some circumstances, while other circumstances, the expert knows that the system can't be trusted, and this difference is one that can't be explored with the STE with the unavailability of expert subjects.

There are also limitations regarding the IMPRINT model constructed. One of the limitations is that a single model can't account for different strategies that different people may use. The IMPRINT model is also limited in that it does not take into account learning effects that occur from a subject seeing similar types of alerts multiple times.

Section 1.6: Implications

By demonstrating that it's possible to use workload analyzers to predict the impact of manipulating factors in Human Factors experiments, it's possible to improve experimental design across the Human Factors domain as a whole, leading to more effective experiments. By using IMPRINT in conjunction with the STE, it's possible to identify specific causes of bottlenecks in performance, and examine the specific drivers of workload within a task.

Preview

Chapter 2 is the literature review, which explores the work done thus far to understand the cognitive demand that network analysts face, along with work done to create simulated task environments for different domains. Chapter 3 details the methodology, which elaborates on the design process to create this simulated task environment, including the task analysis done on the network analysts, and interaction with the researchers that may use this task environment to ensure that it can meet their needs. Chapter 3 also describes the creation of a simulation of the task in the workload analyzer IMPRINT, and how it can be used to identify the settings of factors that would properly drive workload and performance to the degree that they would span the breadth of interesting workloads. Chapter 3 also describes how an experiment could be conducted to demonstrate how this task environment can be used to demonstrate that these factors are able to replicate various levels of workload and performance. Chapter 4 contains the results and analysis of an experiment done with IMPRINT to identify conditions that can be used with an STE Calibration Study. Chapter 4 also contains the results and analysis of the STE Calibration Study. Chapter 5 presents an evaluation of the task environment, and the IMPRINT model, and shows how it can be used to improve the capabilities of researchers and network analysts, along with providing suggestions for future work.

II. Literature Review

Chapter Overview

The purpose of this chapter is to illustrate the research previously performed to understand the work of network analysts along with research done along the lines of Simulated Task Environment design, and Cognitive Task Analysis as a whole.

Section 2.1: Network Analysts Research

A great deal of work has been performed to document the various tasks within the Cyber Domain and the challenges associated with each task. The National Initiative for Cybersecruity Education has developed a framework that describes the breadth of jobs across the domain (National Initiative for Cybersecurity Education, 2011). This framework provides a high level overview of the different types of tasks that each job performs, along with the knowledge skills and abilities required for each. A common theme across this domain is some type of analyst attempting to understand large amounts of data and respond to malicious activity. The large amount of data may lead to the human operator being overloaded with too much information at once. This information overload can lead to difficulty in performing the task the operator is assigned (Voorhees, 2007). There has been a large amount of research that seeks to understand the analysts' tasks and construct better interfaces that allow the analysts to maintain a higher level of situation awareness.

One of the many different types of analysts is Network Analysts. Network Analysts monitor network traffic for suspicious activity and act accordingly. D'Amico and Whitley examine the different types of Network Analysts, and the tasks they perform

(D'Amico & Whitley, 2007). These types of Network Analysts include Line Analysts, Escalation Analysts, and Correlations Analysts. Line Analysts perform the first look at the data coming in, determining which data requires further analysis. This level of analysis typically takes no more than a few minutes. Then Escalation Analysts investigate the suspicious activity identified by the Line Analysts. This investigation can take up to multiple days and looks at data from a number of different sources. Lastly, the Correlation Analysts look at trends in alerts instead of focusing on individual incidents (D'Amico & Whitley, 2007). Of these, the focus of this research is on the Line Analysts.

Line Analysts must quickly examine the large amounts of data arriving and decide what information is relevant. This information typically comes in the form of alerts, automatically generated logs of suspicious activity. The Line Analysts must quickly identify which alerts require investigation at a higher level. These analysts do not come to a firm conclusion on the data, only identifying activity that is suspicious and should be looked into at a deeper level. The data they use includes information such as packet header information and IDS alerts. They may also use knowledge of suspicious IP addresses to guide their search. Once these Line Analysts have filtered the alerts that require further investigation, this information is passed onto Escalation Analysts (D'Amico & Whitley, 2007). The difficulty Line Analysts face is processing the massive number of alerts quickly, while minimizing the false positives and false negatives they commit (Rice, 2014).

D’Amico and Whitley further examine the requirements for network analysts to carry out their mission effectively. Line Analysts require tools that allow them to identify the data that needs investigated. Removing needless information can be accomplished through automated filters that remove irrelevant data, or visual cues to guide the analysts to the information of interest (D'Amico & Whitley, 2007). A real world example of a visual cue is how ArcSight uses colors to indicate the severity of an alert. Red indicates an alert of high severity, while green indicates an alert of low severity. These colors allow the analyst to see the severity of the alert at a glance and prioritize their tasks accordingly (Voorhees, 2007). D’Amico and Whitley also suggest that color could be used to indicate if an alert has been examined or not, something that could be used to allow these analyst to perform their job quicker as they can easily see which alerts need investigated and which ones do not. The analysts are also required to identify the key information they need quickly, meaning that any tools built for them need to consider how it helps the analysts identify the key information (D'Amico & Whitley, 2007). Correlation Analysts oftentimes introduce filters to preventing known non-threats from displaying. However, as the types of threats change frequently, the filters that may prevent a majority of non-threats one day, may do little the next (Stone, 2015).

Line Analysts at different organizations all perform their tasks slightly differently. This research focuses on the 33rd Network Warfare Squadron and the work its analysts perform. These analysts use the software ArcSight to evaluate alerts containing the possibly suspicious activity (Rice, 2014). ArcSight is a commercial product distributed by HP with the purpose of combining logs from various sensors and presenting them to the user in a way that allows them to prioritize alerts of high importance (Hewlett-Packard Development Company, 2012).

Sohail examined how ArcSight works for a small network. He illustrates how ArcSight compiles alerts triggered by computers, such as failed logins, which could be used to deduce that someone is trying to crack a password. However, Sohail's research is on a very small network of a few computers, a contrast of the very large scale of the way the 33rd NWS uses ArcSight. Nevertheless, Sohail's research shows how ArcSight can be used and the capability it provides (Sohail, 2014).

Voorhees discusses a process for working with ArcSight at the configuration level, setting up rules and filters to limit the amount of irrelevant information displayed for the analysts (Voorhees, 2007). Voorhees's work provides valuable insight into how ArcSight is used, and actions that could be taken to reduce the risk of the analyst being overloaded. There are many alerts created from common benign events. There are also times were an alert will be generated falsely; one example Voorhees gave was ArcSight reporting a malicious program had been installed on a computer when it had only been mentioned in an email. Voorhees also notes that ArcSight can't display encrypted traffic, but the meta-information about it, such as its source, destination, and size, may still prove valuable to the analyst.

Voorhees also discusses how the internet, and Google specifically, are credited with being valuable sources of information for analysts as there are oftentimes gaps in knowledge regarding network events. There are times when an analyst may not know about a specific type of application, or the documentation may be incomplete, so the analyst will use the internet to find the needed information (Voorhees, 2007).

Section 2.2: Cyber, STEs Already Used, and Overarching Needs

STEs have been used across a number of different domains. The versatility of many STEs allows them to be tailored to answer specific questions based on the needs of the researchers. Some of this research uses STEs to ask broad questions, where one task may be explored in such a way to provide answers that extend beyond that task. Other research has used STEs to focus specifically on the aspects of network analysts.

Giacobe discusses the difficulty of conveying necessary information in a way that network analysts can easily understand it (Giacobe, 2013). Through his research, he compares using a graphical interface, to an interface of only text, showing that the graphical interface allows analysts to perform their task more effectively. Giacobe's research sought to use simulations to study and understand the work that real analysts must do, and specifically, how different interfaces affect the analysts' situation awareness. His research also sought to leverage the power of STEs to perform research on subjects not familiar with the cyber domain while still getting results that could be applied to the real world (Giacobe, 2013).

Champion et al. have researched how information overload can lead to a degradation of performance. Their research illustrated how situation awareness lessens as the demands of the cyber defense task are increased. This lack of situation awareness translates into threats being misclassified, which in the real world, would lead to security violations (Champion, Rajivan, Cooke, & Janwala, 2012).

The danger of security violations illustrates the need for work to be done to allow these analysts to perform their job effectively in the face of information overload. However, these researchers rely on Rajivan's CyberCog task environment (Rajivan, 2011). CyberCog was made to simulate the real work of an analyst, which makes it difficult for those outside the cyber domain to be able to be used as participants (Giacobe, 2013). CyberCog also provides a more complicated task than the narrow focus of the 33rd

Network Warfare Squadron's analysts, since CyberCog has participants perform a number of different tasks beyond the monitoring for threats that the 33 NWS does.

Another example of an environment constructed to emulate the work of network analysts is idsNETS, which seeks to study the cognitive demand of analysts (Mancuso, Minotra, Giacobe, McNeese, & Tyworth, 2012). However, this environment models the task as a resource management task. IdsNETS was built off of the already existing resource management STE, NeoCities (Hellar & McNeese, 2010), which was used as an emergency response system, so doesn't fit the task model of the Line Analysts in our research. Resource management tasks have a participant in control of a limited number of resources, where the participant is instructed to allocate these resources to deal with some type of situation. In the example of NeoCities, the participants controlled emergency vehicles. Contrastly, the 33rd NWS's task involves investigating alerts, and there aren't any resources the analysts must allocate to perform the task.

Fink et al. explores the task of creating a better interface for those investigating incidents, illustrating the challenges and dangers associated with these visualization tools. Many analysts are reluctant to use tools that they don't trust, afraid that they will impede more than aid in their tasks. The lack of trust illustrates the need to truly understand the information analysts' needs along with the thought process they must go through in order to perform their task, and design interfaces according to this information (Fink, North, Endert, & Rose, 2009).

Finomore et al. have done research that shows how misleading or missing information can impede analysts' effectiveness (Finomore, et al., 2013). Their research focused on a team-based task, where information was split among the team. The information provided to the subjects varied in usefulness, something that is also true for network analysts as many of the alerts they view are non-threats, so they must filter out the irrelevant, and sometimes misleading, information. Their research indicated how misleading information disrupted the team decision making process, which is something Network Analysts also have to deal with.

Brehmer and Dorner provide an early illustration of the tension between laboratory research and field research, with laboratory research lacking the relevance and complexity of the real world, while field research is hindered by a lack of experimental control (Brehmer & Dörner, 1993). They sought to create computer simulated macroworlds, computer programs that would emulate complicated environments, and could be used to study broad concepts by simulating a specific domain. These macroworlds illustrate an early variation of an STE, attempting to bridge the gap between the real world, and a laboratory environment.

Other researchers have benefited greatly from using STEs to replicate a certain task they wanted to explore and to enhance their research. For example the Multi-Attribute Task Battery (MATB), developed by NASA was a task environment to replicate the tasks performed by airplane pilots (Comstock & Arnegard, 1992). It has since been used by researchers to study the effects of sleep deprivation on subjects' abilities to perform a task (Caldwell & Ramspott, 1998). MATB has also been used by Wilson and Russell to study how workload manifests itself through physiological signs and demonstrated that it was possible to predict workload from these physiological measurements (Wilson & Russell, 2003). These studies illustrate how researchers have been able to use STEs in the past to aid in their research.

The RoboFlag STE was able to study how participants handle information overload (Guznov, Matthews, Funke, & Dukes, 2011). The authors illustrate that increased cognitive demands drive a participant’s workload, along with their task engagement, distress, and worry. An interesting relationship they picked up on was how uncertainty can lead to increased task demand. This relationship can tie into how Network Analysts may at times be uncertain about the action they need to take due to not having all the information needed to make a decision.

Galster and Bolia discuss the benefits of using STEs to study the way different interfaces affect subject performances, and, suggest ways to improve interface design for real world products (Galster & Bolia, 2005). Galster and Bolia also discuss the dimensions of STEs that make them useful in different capabilities: tractability, realism, and experimental control. Tractability refers to the STE’s capability to answer the questions the researcher may pose, and also the space on the theoretical vs. applied scale the STE occupies. Realism refers to how much the STE matches reality. The authors point out that low levels of realism answer more broad fundamental questions, while highly realistic STEs are designed to answer much narrower questions with more field validity. Lastly, experimental control refers to the amount of variability the researchers can introduce into an STE. High levels of experimental control can allow the researcher to tailor the task to their needs; however it could also break away from a realistic environment (Galster & Bolia, 2005).

Through a study of automatic decision support systems, Galster and Bolia also specified that they needed a “real time metric of decision quality” (Galster & Bolia, 2004). This metric requires that an effective STE must also provide a real-time response of right and wrong decisions. Their research also acknowledges that the complexity of an STE is significantly less than a real world task; however the lessons learned can still be applied to reality, or guide future research (Galster & Bolia, 2004). Cooke and Shope describe the general goal of any STE to reproduce the behavior and cognitive processes of a real world task within a laboratory setting. By having the task within a laboratory setting, experimental control can be applied and measurement capabilities still exist. (Cooke & Shope, 2005).

There has also been work done to document the process of creating a good STE. Cooke and Shope explain their process for creating a STE that was made to emulate the task of controlling Uninhabited Air Vehicles (Cooke & Shope, 2004). The authors describe the benefits of using STEs. The benefits include being able to use the STE as a high fidelity simulations, that allows researchers to examine the task, without needing to fully replicate the task. They explain how the goal of an STE is to recreate the task in a way that performing the task requires the same thought processes, even if it does not have the same “look and feel” of the actual task. A good STE will also be tailorable for research purposes, allowing those using the STE the flexibility and tools they need, perhaps at the cost of fidelity. Consequently, a good STE will both provide experimental control and the necessary representation of the environment being studied.

Cooke and Shope break their process to design their UAV STE, into 4 steps. The first step is to gather information about the task. This step includes narrowing the design to a single representation of the task. They choose the Air Force Predator to be their focus to study. They also examined the research objectives the STE is aimed to meet. In their example, they needed an STE which would lend itself to team cognition research, something not all STEs require. Information also needs to be gathered about other constraints that could limit how an STE could be designed--for example, budget concerns or computational power. The second step that Cooke and Shope describe is abstracting features. This is the step that decides how certain features of the real world will be replicated in the STE. During this process, some parts of the task may be exaggerated, while others may be omitted, depending on the goals for the STE. The third step is to build a prototype. This step starts by using a very low tech model to give a picture of what the final product will look like--in their case, a paper mock up with some simple drawing software. This process allows them to get a quick and cheap picture of what the task environment will be. As they get feedback from researchers and subject matter experts (SMEs), they fine tune their design. Once their design is in place, they translate the visual design into a software design. The software design considers the data that needed to be stored, the variables to be manipulated, and the communication between units. Once the design step is completed, step 4, implementation can begin. This step includes a cycle of interactions where programmers would build the STE, the researchers would provide feedback, and then the programmers would make the required improvements. This cycle led to a product that was able to meet the researchers' needs. Cooke and Shope also describe lessons other STE designers should keep in mind, such as designing for future expansion and considering required tradeoffs. The major contribution Cooke and Shope make is outlining the steps to create an STE, something that other researchers can and have built upon (Cooke & Shope, Designing a Synthetic Task Environment, 2004).

Section 2.3: Cognitive Task Analysis

Cognitive Task Analysis (CTA) is the process of understanding a task and the underlying cognitive activity that the task requires. This cognitive activity can include decision making, mental calculations, and problem solving. Some questions that frame a cognitive task analysis are:

- What issues will be addressed?
- What product will be delivered at the end of the project?
- What people can explain an issue?
- What types of cognition need to be understood?
- What situations provide the most information about the issues being explored?

Another important aspect of cognitive task analysis is observing real work being done, as opposed to just asking questions of the SMEs. This real world perspective can allow researchers a much better understanding of the task in question than simply conducting interviews (Crandall, Klein, & Hoffman, 2006).

Crandall, Klein & Hoffman also discuss that a key part of talking with SMEs is getting them to tell stories. While general descriptions of the SME's work are necessary to understand the task being investigated, it is the stories that highlight real events, and allow the investigators to see how the pieces of the task come together in a real world event. Stories also highlight memorable events, those that while possibly outside the realm of normality, may be of particular interest to investigators. These extreme events can show what techniques and knowledge were used and how the SMEs resolved the situation. The stories can also highlight the abilities that an experienced person may have over an inexperienced one, the cues they can pick up on, so these stories can serve as a learning opportunity. The stories can also highlight the kinds of decisions an expert may have to make, along with what makes these decisions difficult. Their research provides valuable insight in how a task analysis can be done for our research (Crandall, Klein, & Hoffman, 2006).

Summary

Thus far, there has been a great deal of work done to understand the cognitive demands of cyber analysts, as seen from the analysis work done by D'Amico and Whitely (D'Amico & Whitley, 2007), along with the experiments done with Mancuso's idsNETS (Mancuso, Minotra, Giacobe, McNeese, & Tyworth, 2012) and Rajivan's CyberCog (Rajivan, 2011). There has also been a large amount of research done creating STEs for various domains. Cooke and Shope provide a valuable insight into the features an STE needs to provide for researchers (Cooke & Shope, 2004). Galster et al. illustrate how STE demands may fluctuate based on the demands of the task being researched (Galster & Bolia, 2005). Crandall, Klein & Hoffman provide insight for how to perform cognitive task analysis, by illustrating techniques to get subject matter experts to explain the aspects of their tasks that are valuable to researchers (Crandall, Klein, & Hoffman, 2006). The role this research seeks to fill is to leverage the knowledge gained from these previously mentioned works to create a STE for a Network Analyst performing the type of role that those of the 33rd NWS perform, something that has not be done before. In addition a new contribution of this research seeks to demonstrate how IMPRINT can be used to model that task emulated by the STE. This model will make it possible to predict how a subject would perform and how their workload would change based off the factors manipulated.

III. Methodology

Chapter Overview

This chapter presents the overarching goal of the research, to present a new way to perform Human Factors research, by using Workload Analysis software such as IMPRINT to guide experimental design. By using IMPRINT it's possible to simulate dynamic conditions for a subject, and identify the resulting changes in workload and performance. By identifying these changes in workload and performance, it's possible to understand what effects various factors would have on a real subject. However, using IMPRINT in such a way requires an intricate knowledge of the task to be performed, including how each small subtask contributes to workload, along with what success and failure mean at every action taken. The relationships between the conditions and performance may vary based on the specific task and the person performing the task, but identifying these relationships is something that can be more cost effective than running a pilot study with real subjects.

The end goal of this research is to create a system where an IMPRINT model and STE experimental design are able to constantly refine each other, to the degree that IMPRINT is able to accurately model the thought process of a human while correctly modeling the timing and accuracy of all subtasks. An accurate model makes it possible to know what kind of effects changing conditions would have on subjects which aids in experimental design.

The chapter is composed of seven sections. Section 1, Cognitive Task Analysis, discusses the research done to understand the work done by Network Analysts. Section 2, Development of The Synthetic Task Environment, includes the research done to understand the needs of researchers who may use the STE and the design and implementation of the STE. Section 3, IMPRINT Modeling, presents using IMPRINT to model the task of the analysts, something that is done first by modeling the steps of the analysts' tasks in IMPRINT, and then populating IMPRINT with workload and timing data for each subtask, along with what the impacts of success and failure are at each subtask. Section 4, IMPRINT Experiment describes an overview of an experiment done with the IMPRINT model, Section 5, IMPRINT Evaluation Criteria describes how the results of the IMPRINT Experiment will be analyzed and what the results are expected to look like. Section 6, STE Calibration Study Protocol, describes how an experiment could be done with the STE based off the information obtained from IMPRINT with the goal to verify and refine the IMPRINT model. Section 7, STE Calibration Study Evaluation Criteria describes how the results of the STE Calibration Study will be evaluated.

Section 3.1: Cognitive Task Analysis on Line Analysts

Of the steps outlined by Cooke and Shope for designing an STE (Cooke & Shope, 2004), the first step is understanding the task that needs to be emulated. This step includes narrowing the target domain to a specific task, and once the specific task has been identified, learn from literature and subject matter experts (SMEs) how this task is performed.

The process of understanding the task performed is accomplished through a combination of literature research, and interviews with Subject Matter Experts (SMEs), those who have experience with the target task. In order to learn about the process the

Line Analysts use, it is necessary to seek those who had previously and currently worked either with, or as, Line Analysts. A broad overview of the domain is needed, something that can be obtained by interviewing those with a general knowledge of the task. Once the task of interest is narrowed, then those who are closely connected to the task can be sought, which in the case of this research, are those who currently work for the 33rd NWS. By interviewing these individuals, it's possible to get firm answers about the task in a way that the task can be emulated. This process isn't composed of a single interview with each party, but an iterative process, as an answer to one question may lead to more questions that aren't obvious at first. These interviews go beyond just the general overview of the task itself, with the goal to be how the SMEs think and the decisions they must make. Crandall, Klein and Hoffman discuss different ways to frame these interviews, from getting the interviewees to tell stories about notable experiences, or to frame their knowledge in a concept map (Crandall, Klein, & Hoffman, 2006). This process makes it possible to understand not just the steps that the analyst take in carrying out their task, but the reason why they take each step, and the ways various pieces of information come together in their minds.

From the interviews with those who were involved with the 33rd NWS, it is possible to gain an understanding of the task they performed. ArcSight is used by the 33rd NWS to monitor network traffic at the boundary of the network between the Air Force and the public internet. ArcSight monitors the traffic for anything that could be considered suspicious (Rice, 2014). ArcSight is not a defense system itself, but is instead a compilation tool, taking in information from various sensors on the network, such as e-mail servers, DNS servers, and Intrusion Detection Systems. ArcSight then displays these suspicious activities as alerts. Each alert is displayed with a small amount of information on a constantly scrolling display. One of the most relevant pieces of information is ArcSight's estimate of the severity which is on a scale that includes Very Low, Low, Medium, High, and Very High. The severity helps the analyst decide which alerts require the most immediate attention. The severity is displayed as a color, making it easy for the analyst to see at a glance how severe the alert is (Lovell, 2014). Based on this information, if the analyst determines the alert requires further information, they will click on it, which will bring up more detailed information (Rice, 2014).

While it may vary across organizations, in the 33rd NWS, the analysts do not act directly on an alert that they deem important, instead they send the alert to the incident response team (IRT) for further investigation. This means there are two main decisions the analysts must make. The first decision is to decide what alerts should be selected for investigation, and the second decision is to decide from that investigation, which alerts should be sent to the IRT. Figure 1 illustrates this process.

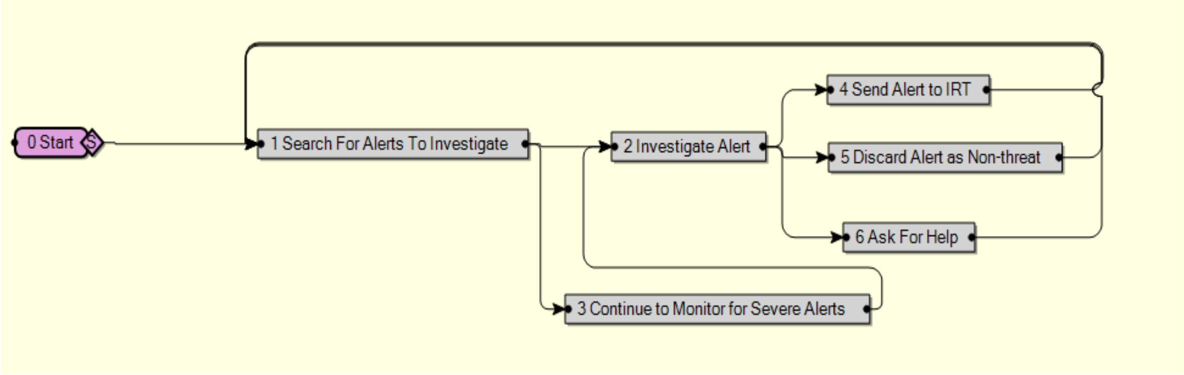

0 Start
S
1 Search For Alerts To Investigate
2 Investigate Alert
4 Send Alert to IRT
5 Discard Alert as Non-threat
6 Ask For Help
3 Continue to Monitor for Severe Alerts

Figure 1: 33rd NWS Analyst Process

Ideally, all alerts would be able to be investigated, to ensure that nothing that was a potential threat went by undetected (Lovell, 2014). However, the sensors that generate alerts do so in a way that creates many alerts that aren’t actual threats, creating an infeasible amount of work for the analysts to do. Therefore, analysts typically only investigate alerts classified by ArcSight as Medium or higher, and those alerts that correspond to these more severe alerts as a way to make their task load more manageable (Lovell, 2014). In the most extreme cases, the ratio of true threats to alerts could be worse than 1:1000. In some cases, analysts may use the filters within ArcSight to reduce the non-threats shown, but even with these reductions, there may still be too many alerts for the analysts to handle (Hannan, 2014). These filters are rarely developed by Line Analysts, but instead by the Escalation Analysts after seeing a continual pattern of easily filtered non-threats (D'Amico & Whitley, 2007). Because the filtering is not done by the Line Analysts, a Simulation Task Environment made to emulate the work done by a Line Analyst, doesn’t need to consider filtering.

The end goal for the Line Analyst is to mark all alerts as either non-malicious or reportable. Some alerts can be deemed to be non-malicious by just what the alert initially displays, for example the type of alert might be known to always be a non-threat, or the sensor could have already handled the event such that no further action is needed. Other alerts require the analyst to investigate the alert to get more information in order to make a decision. Typically, the most important factor in choosing which alerts require investigation is the severity. In some cases, less severe alerts are also investigated if something is known to be suspicious in them, or they are connected to a more severe alert. On the other hand, some of the alerts that would typically require investigation may be known to be able to be ignored. For example, ArcSight may identify an attack that would normally be a threat, but if the vulnerability was known to be patched, then the alert could be safely ignored. The analysts may also tend toward investigating alerts they are familiar with, as opposed to investigating an alert they have little knowledge about.

When an alert is clicked on, the analyst receives all the information the sensors sent to ArcSight. This information can include session information, a pcap[1] (packet capture) file and more specific information varying based on the type of alert. With this information, the analyst will decide if the alert is malicious (Hannan, 2014). This process varies depending on the type of alert, but will often involve examining the IP addresses along with any website names provided to determine the reputation of the cause of the alert. Analysts will often examine the signature that caused the alert, and attempt to find it in the pcap file. Depending on the experience and knowledge of an analyst, for familiar types of signatures, they may not need to look it up to know what to look for. Based on experience and knowledge, some analysts may also not need to look up an IP address to have an idea of where it is coming from.

In a majority of cases, the analyst will discover that the pcap only partially matched the signature, and can therefore classify it as a non-threat since the alert doesn’t represent the signature (Lovell, 2014). Depending on their experience with the type of alert, the analyst may ask other analysts for help in understanding the information before coming to a conclusion. If the analyst determines the alert is a non-threat, they will close the alert, a process that includes adding a comment in ArcSight that the alert is non-malicious along with any required details. If the analyst determines the alert is malicious,

[1] A packet capture file shows all the network packets that went through the network. This can be viewed down to the binary level, but more abstract views are often provided showing the information at each network layer in a human-readable format.

they will indicate this in an ArcSight comment, and write a report which is sent to the IRT (Lovell, 2014; Hannan, 2014). There are also some cases where the analyst can't come to a decision. This occurs when there's not enough data to say the alert is definitively worth flagging as a threat, but enough evidence that it can't be discarded entirely. In these cases the analyst may just leave it, though include a comment that the analyst will continue monitoring it and keep it in mind if anything in future alerts may provide more information about it (Lovell, 2014). Notes from all interviews performed are provided in Appendix E.

Section 3.2: Development of the Synthetic Task Environment

The process of creating the simulated task environment is composed of four steps, similar to the steps described by Cooke and Shope (Cooke & Shope, 2004). The first step is to understand the task that needed to be emulated, which was described in the previous section. Second, the requirements for the STE must be considered. In order to identify the requirements, it is necessary to consider who the intended audience is, which in the case of this research is the 711th HPW. The audience determines the requirements, both in the complexity of the task to be emulated, and the capabilities the STE requires. The third step is to design and build the STE. The design process is an iterative process that encompasses verifying design ideas with SMEs, and the target audience, along with involving those who will be implementing the software to ensure all requirements are firmly understood and are feasible giving the time and resource constraints. Finally, the STE needs to be evaluated both to ensure that it is a working program and that it meets the requirements identified in the first two steps.

The analyst’s task can be broken up into three steps: identifying the alerts that require investigation, investigating these alerts, and identifying which of these investigated alerts represent a true threat, thus requiring further action. A STE representing this task must emulate these steps. The simulation needs to show the alerts scrolling by, allowing the user to bring up more detailed information by clicking on an alert. Once the detailed information is brought up, the user must then be able to make a decision based on the information. A key part of an STE is abstraction, simplifying the technical details of the task while still maintaining the cognitive decisions that need to be made. This abstraction makes it so that a novice user could still use the simulation with a small amount of training and be able to perform the task adequately (Cooke & Shope, 2004). Abstraction can be done through simplifying the technical depth of the information presented. For example, the information the in the pcap files, the complexity of the various network protocols, and the information regarding various possible threats all need to be simplified. The tools the analysts use may also need to be abstracted, such as if multiple tools are used for the same type of task, it may be possible to combine them into a single tool.

The STE also needs to meet any requirements set in place by its target audience. The target audience could be a specific customer that wants the STE to perform a specific role in a single experiment, or it could be a broader situation where the STE is hoped to be useful in a variety of circumstances. The target audience can affect many things, such as the abstraction required, the customizations offered, and the data recorded. For this research, the STE is expected to be used by Human Factors researchers in the Human Performance Wing (HPW) of the Air Force Research Laboratory (AFRL). In order to meet the needs of these researchers, the STE must keep track of choices the subject makes and provide a level of abstraction such that a college student--with no knowledge of the Line Analyst task--can be used as a test subject. In addition, the STE needs to record any data that may aid in the analysis of the experiment, as well as provide the needed customization and extendibility to fulfill the needs of various possible experiments.

The choices subjects make is one of the key pieces of information that would show their thought process. As such, the STE needs to show when they decide to investigate the alert, and when they decide whether it’s a threat or not. It’s also important to be able to identify the difference in a subject ignoring an alert because they didn’t see it, as opposed to ignoring it because they know it’s not a threat, and as such no action needs to be taken. In order to accomplish this differentiation, this STE includes a feature where the subject can right click on an alert and select the option to flag as a non-threat without needing to investigate the alert. While this feature does not exist in the actual task, knowing why a subject ignores an alert is a valuable piece of information for researchers.

Beyond just knowing the times that these decisions are made, the STE also needs to keep track of the subjects’ mouse and keyboard actions, so the researcher can see what the subject was doing at any time. Customization is also needed to allow the researcher to adjust as many of the aspects of the STE as possible. These customization options include if the subject can mark an alert as a non-threat without investigating it, along with the type of data shown during the investigation, which allows the researcher to decide how realistic and complex they want the simulation to be. For example, instead of displaying an Internet Protocol (IP) address as a series of 4 octals, the researcher may just want it to be a decimal number from 0 to 1000 to make it easier to understand. These customization options circumvent some parts of the risk of the STE being designed to be too abstract or too complex.

The use of the STE also needs to be fairly intuitive. While it can be assumed that the researchers have some degree of technical knowledge, and can spend more time learning how to use the STE than the subjects, making it understandable how to use the various customization features is also important. The output of the STE should also be intuitive. For example, the STE should make it simple to figure out which alerts the subject flagged correctly versus incorrectly. The STE should also make it easy to identify the actions the subject took to come to a conclusion about a certain alert.

With the tasks identified, it is possible to begin designing and constructing the STE. This stage includes making various software design decisions, and depending on the scale of the project, the design decisions may involve meeting with programmers. One of these decisions was the programming language to be used, something that is determined based on programmers' expertise, and how built in functionality corresponds with the various needs for the STE. For this STE, the C programming language was used to create the STE, as there were libraries that provided the graphical capabilities to construct the STE. Another important factor to consider is how the STE obtains its information, such as the script for when events occur.

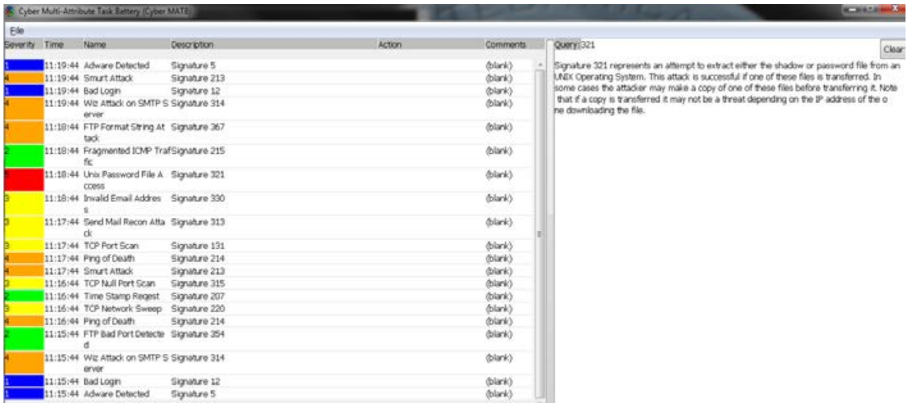
In the case of this STE, all the input is provided with csv[2] files. One csv file includes all the alerts that enter the system during the trial. Other csv files provide the detailed information about the alert when it is investigated. Another csv file provides responses for when the subject performs a query.

The details of the output of the STE also need to be specified. The output of the STE needs to provide all the information the researchers could need for their analysis. One required part of the output is a timestamp for whenever a subject performs an action. The actions include investigating an alert, making a decision on an alert, or querying for information. The timestamps need to be in terms of time since the trial started, or have the trial start time also recorded so the time since the start of the trial can be calculated. The timestamps make it possible to understand when the subject makes decisions, and how much time it takes for them to investigate each alert. Another piece of required information is the final decision made by the subject for each alert. By knowing the decisions the subject made, it's possible to calculate how well the subject performed. Lastly, the STE needs to record mouse and keyboard actions. By capturing the mouse and keyboard actions, it's possible to replay the trial.

The design process is iterative, as many of the clarifications needed for implementation require various design decision to be made, which may then add more questions. In some cases, the design process also requires clarification from the SMEs to ensure everything is as realistic as possible, and when realism is abandoned, there is a sufficient justification for this. A more detailed version of the STE design for this research is included in Appendix A.

[2] A csv file is a text file with values separated by commas, which gives the values a table like format

Once the details of the STE design are finalized, the STE can be built. Figure 2 below shows a screen shot of the STE.



Severity	Time	Name	Description	Action	Comments
1	11:19:44	Adware Detected	Signature 5		(blank)
4	11:19:44	Smurt Attack	Signature 213		(blank)
1	11:19:44	Bad Login	Signature 12		(blank)
4	11:19:44	Wiz Attack on SMTP Server	Signature 314		(blank)
4	11:18:44	FTP Format String Attack	Signature 367		(blank)
2	11:18:44	Fragmented ICMP Traffic	Signature 215		(blank)
5	11:18:44	Unix Password File Access	Signature 321		(blank)
3	11:18:44	Invalid Email Address	Signature 330		(blank)
3	11:17:44	Send Mail Recon Attack	Signature 313		(blank)
3	11:17:44	TCP Port Scan	Signature 131		(blank)
4	11:17:44	Ping of Death	Signature 214		(blank)
4	11:17:44	Smurt Attack	Signature 213		(blank)
3	11:16:44	TCP Null Port Scan	Signature 315		(blank)
2	11:16:44	Time Stamp Reqest	Signature 207		(blank)
3	11:16:44	TCP Network Sweep	Signature 220		(blank)
4	11:16:44	Ping of Death	Signature 214		(blank)
2	11:15:44	FTP Bad Port Detected	Signature 354		(blank)
4	11:15:44	Wiz Attack on SMTP Server	Signature 314		(blank)
1	11:15:44	Bad Login	Signature 12		(blank)
1	11:15:44	Adware Detected	Signature 5		(blank)

Signature 321 represents an attempt to extract either the shadow or password file from an UNIX Operating System. This attack is successful if one of these files is transferred. In some cases the attacker may make a copy of one of these files before transferring it. Note that if a copy is transferred it may not be a threat depending on the IP address of the one downloading the file.

Figure 2: STE Screen Shot

After the STE is completed, it must be evaluated. Evaluation includes ensuring that all the functionality works properly. In addition, the evaluation also includes error checking, making sure that the STE responds adequately if the subject acts in an unexpected way. Testing also needs to span the breadth of requirements that an experiment will need. An example of this testing is to ensure that the system responds correctly when the maximum amount of allowed alerts are in the system and open. The portability of the STE also needs to be confirmed, so that the STE is confirmed to be able to run on different types of computers, such as those with a different operating system or a desktop versus laptop. Lastly, the output of the STE must be checked to ensure that it's recording all the information properly. Once the testing is complete, the STE can then be used for a sample experiment to demonstrate its ability to drive workload at various levels.

Section 3.3: IMPRINT Modeling

IMPRINT is a software tool developed by Alion Science and Technology on behalf of the Army Research Labs (ARL), Human Research & Engineering Directorate. IMPRINT simulates a network of discrete tasks. Its purpose is to model human performance, both across a team, and also for a single operator throughout a mission. It can also create workload profiles, making it possible to see how workload is distributed across a team, and also how task allocation affects a single operator (Army Research Laboratory, 2010).

IMPRINT has the capability to import activity diagrams from Microsoft Visio, but it is also possible to manually create a task network in IMPRINT. Thus, the first step of using IMPRINT is to take the activity diagram constructed from the task analysis, and to recreate it in IMPRINT. IMPRINT's structure has the task organized like a tree, where a subtask may be a leaf node, with no other subtasks under it, or a function, that is composed of smaller subtasks. By building the task network with IMPRINT, it is possible to see any deficiencies in the activity diagram, such as redundant or missing substeps. Once the activity diagram is in IMPRINT, then it's possible to start assigning workload values to the task nodes.

For the purposes of this research, only a single human operator is being studied. Only studying a single operator simplifies the requirements of the IMPRINT model, only needing to track a single human operator instead of multiple operators.

What makes designing IMPRINT different than designing the STE is that IMPRINT is human-centric, where the nodes in IMPRINT primarily represent where the human's attention is focused at any given moment. This is a contrast to the event-centric nature of the STE, where the focus of design is keeping track of alerts and related information. The human-centric nature of IMPRINT gives it a great amount of power, as it can examine the human specifically, identifying what tasks the human is performing, and their workload at any moment in the trial. However, focusing on the human's thought process adds a layer of difficulty, as the alerts are also flowing through the system. A challenge of using IMPRINT is keeping track of the data that represents the alerts, as the data within each alert affects the decisions the operator needs to make. The data needs to be easily accessed throughout the entire system, along with being stored in a way that's easy to work with. There are different options to handle this data, each with their own challenges and benefits. The solution that is chosen needs to allow the IMPRINT model to access the needed data whenever a decision needs to be made, while also being simple to implement. In addition to the simple implementation being easier to implement, it also makes it easier for the model to be expanded as well, especially if someone else is the one expanding the model.

Five different options are considered to keep track of the alert's data. The first option is to create an Alert class, and then have each alert represented by an object. The second option is to have an array for each alert, where each field in the array stores a specific piece of information. The third option is to store the information in an external csv file. The fourth option is to store the information that would be in the csv in a String variable within IMPRINT. The fifth option is to not store information about each individual alerts, but instead to store a count of how many alerts of certain categories are in the system.

The first option, creating an Alert class is a natural application of the Object Oriented paradigm. It is possible to easily access and modify the data for each alert. IMPRINT supports C# code snippets, and C# supports Object Oriented programming, however it is impossible to create Classes within IMPRINT, since IMPRINT only implements a subset of C# in its code snippets. IMPRINT does allow plug-ins to be added, which make Objected Oriented programming possible, but requiring the use of plug-ins requires an additional layer of complexity.

The second option, having each alert's information stored in an array, is similar to the first option. The difference is that instead of using classes which require plug-ins, all the information is stored in arrays which IMPRINT's code snippets support. The problem with this option is that using arrays in such a way requires a convoluted conversion between the support objects provide, and the limited ways to use the data in an array. This problem makes expansion difficult, and could easily lead to code readability problems.

The third option is having the information stored in an external csv file. Having the information stored in an external file makes it easier for the researcher to configure the types of alerts that enter the system. The csv files can be created with external scripts or programs, and then tailored to researcher needs. The problem with this option is that while C# does support reading data from a file, the subset of C# that can be used in code snippets does not. The lack of support means that plug-ins need to be used, adding an additional layer of complexity. An additional problem is that the information needs to be stored somewhere once the information is imported into IMPRINT from the csv, meaning that this option would need to be combined with one of the others to work.

The fourth option is using a String within IMPRINT to store all the information that would have been in the csv file. This option circumvents the problem of using plug-ins. One of the problems of this option is the fact that a massive amount of information is stored in a single string which make the data difficult to work with. Another problem is that the entire string will likely need replaced with each run of IMPRINT, creating a large overhead for when IMPRINT experiments are performed.

The fifth option is using counts of categories of alerts instead of storing each alert on its own. In this option, IMPRINT will keep track of the number of “Very High” alerts that are in the system, but won’t have all the information about each of those alerts. The information not stored by the categories the alert is in is determined randomly based on probabilities defined in other IMPRINT variables. The benefit of this solution is that it can be done entirely with functionality supported within IMPRINT’s code snippets. It also only needs integer variables to keep track of the counts for each category, and variables for the probabilities. A challenge of this solution is that care must be taken to ensure that all the categories are updated correctly. Other variables may need introduced to differentiate between alerts just in the system, those currently being investigated, and the alerts where an action had been taken on them.

For this model, option 5, using categories, is chosen to be implemented. This option requires no plug-ins to work, and uses all types of variables in a way that is intuitive. The variables that are needed to keep track of the alerts do require some additional implementation overhead; however the additional variables help to illustrate how the alerts flow through the system.

The categories are used to keep track of the severity of each alert (Very Low, Low, Medium, High, Very High), along with if the alert is a threat or not. Since the categories keep track of if the alert is a threat or not, it’s possible for the IMPRINT model to check if the alert is a threat whenever needed. Whenever the simulated operator makes a decision of if the alert is a threat or not, the model compares that decision with the information known about the alert to see if the operator is right or wrong.

With the model in place, including both the focus on the human’s attention, and the support for tracking the alerts, it is then possible to start entering workload values for the sub tasks. IMPRINT breaks workload into 7 categories, Visual, Cognitive, Auditory, Fine Motor, Gross Motor, Speech, and Tactile. Values for each of these categories are assigned at each of the lowest level sub tasks. These numerical values are not assigned subjectively, but by selecting a short description that most closely matches the task, which equates to a pre-defined, and expert verified numerical value. These predefined values make it so that if two different people assigned workload values across the model, the values would be consistent.

Due to the nature of the task, the only categories of workload that matter for this task are Cognitive, Fine Motor, and Visual. Cognitive refers to the decisions the operator makes, such as if an alert is a threat or not, and the mental processing to come to these decisions. Fine Motor refers to small movements, such as moving a mouse or typing on a keyboard. Visual refers to reading the information about the alerts, or looking at the colors that represent severity to know which alert to investigate next.

The other categories of workload do not apply to this task. Auditory refers to listening for and responding to sounds, something that the task does not currently require the operator to do. Gross motor refers to large body movements, such as walking, which the STE does not require the operator to do. Speech refers to the operator speaking, something the task does not require. Tactile refers to information obtained through the sense of touch, for example, differentiating between a rough surface and a smooth surface, something that the task does not require.

Tables 1 through 3 below show the descriptions and values that IMPRINT uses for the Cognitive, Visual, and Fine Motor, categories.

Table 1: IMPRINT Cognitive Workload Values

Task Description	Workload Value
Nothing	0.0
Automatic (Simple Association)	1.0
Alternative Selection	1.2
Evaluation/Judgment (Consider Single Aspect)	4.6
Rehearsal	5.0
Encoding/Decoding, Recall	5.3
Evaluation/Judgment (Consider Several Aspects)	6.8
Estimation, Calculation, Conversion	7.0

Table 2: IMPRINT Visual Workload Values

Task Description	Workload Value
Nothing	0.0
Register/Detect (Detect Occurrence of Image)	1.0
Inspect/Check (Discrete Inspection/Static Condition)	3.0
Locate/Align (Selective Orientations)	4.0
Track/Follow (Maintain Orientation)	4.4
Discriminate (Detect Visual Differences)	5.0
Read (Symbol)	5.1
Scan, Search, Monitor (Continues Serial Inspection)	6.0

Table 3: IMPRINT Fine Motor Workload Values

Task Description	Workload Value
Nothing	0.0
Discrete Actuation (Button, Toggle, Trigger)	2.2
Continuous Adjustive (Flight Control, Sensor Control)	2.6
Manual (tracking)	4.6
Discrete Adjustment (Rotary, Vertical Thumb Wheel, Lever Position)	5.5
Symbolic Production (Writing)	6.5
Serial Discrete Manipulation (keyboard)	7.0

Once the values for workload are in place, it is necessary to determine the time that each subtask is expected to take along with the expected variance. Before any experiments are performed with human subjects, it may be difficult to determine exact durations for each sub task. IMPRINT provides some built in functionality for determining how long small events such as moving a mouse can take, but events that include cognitive processing information must be estimated.

IMPRINT also requires probabilities set for the chance of failure at any specific sub task, and the associated action to take when this failure occurs. Similar to the times for each sub task, the expected failure rate is difficult to determine until experiments are performed with human subjects, so they must be estimated. Failing a task could be as simple as miss-clicking, something that can be easily fixed after a couple seconds, or it could be misreading a pcap and identifying something as a threat when it isn’t, thus decreasing the performance score.

When running the IMPRINT model, factors can be manipulated to see how workload and performance change with these factors. IMPRINT provides workload values at every time during the trial that the workload changes. From the workload values, a time-weighted average can be taken across the entire trial. The individual workload values can be useful in identifying during what parts of the trial the operator experiences the highest or lowest workload, and may provide insight if a high workload precludes failure. In contrast, the time-weighted average workload gives a higher level view of the trial, and can illustrate how hard the operator worked as a whole. The workload value provided by IMPRINT only reflects the work the simulated operator does. If the operator is given twice the task than the operator is capable of, and the operator performs half the task, then the workload is the same as if the operator is only given the amount of task it’s able to handle.

A full description of the nodes and functions within IMPRINT are provided in Appendix D.

Section 3.4: IMPRINT Experiment Description

For the IMPRINT experiment, two factors are chosen to be manipulated. These factors are the amount of Alerts Per Minute that enter the system (APM), and the distribution of the severity of the alerts.

There are 10 APM levels explored, ranging from 1 to 10, with the APM level representing how many alerts arrive at each minute. At the start of each minute of the trial, a number of alerts enter the system consistent with the APM level.

The alerts are split into 5 categories based on severity, Very Low, Low, Medium, High, and Very High. For the case of this experiment, the higher severity, the more likely the alert represents a true threat. This is a contrast to the real world, where the severity represents the damage caused if the alert is a true threat.

The typical distribution of alerts is provided in Table 4. These values were obtained from an SME (Lovell, 2014).

Table 4: Typical Alert Severity Distribution

Type of Alert	Percentage of All Alerts
Very Low	25%
Low	25%
Medium	20%
High	20%
Very High	10%

Using the SME provided distribution as a middle point, 21 levels of the severity distribution are made. 10 of these levels have a smaller amount of severe alerts than the SME provided distribution, while 10 levels have more. The distribution of the alerts for all 21 levels is shown in Tables 5 below.

Table 5: Severity Level Distributions

Severity Level	Percent Very Low	Percent Low	Percent Medium	Percent High	Percent Very High
1	35	35	20	10	0
2	34	34	20	11	1
3	33	33	20	12	2
4	32	32	20	13	3
5	31	31	20	14	4
6	30	30	20	15	5
7	29	29	20	16	6
8	28	28	20	17	7
9	27	27	20	18	8
10	26	26	20	19	9
11	25	25	20	20	10
12	24	24	20	21	11
13	23	23	20	22	12
14	22	22	20	23	13
15	21	21	20	24	14
16	20	20	20	25	15
17	19	19	20	26	16
18	18	18	20	27	17
19	17	17	20	28	18
20	16	16	20	29	19
21	15	15	20	30	20

By combining the 10 APM levels, and the 21 Severity levels in a full factorial design, there are a total of 210 conditions that IMPRINT will run.

Each of the 210 conditions is run 15 times for a total of 3150 runs. The random number seed is different for each the 15 times the simulation is run. The changing random number seed affects anything that has a distribution, such as the time some of the nodes take. The changing seed also affects anything chosen probabilistically.

Section 3.5: IMPRINT Evaluation Criteria

Each alert that entered the system could be classified into 6 categories as shown in Table 6 below. The rows indicate the true status of the alert, and the columns are the actions that the operator took.

Table 6: Alert Classification Categories

	Alert Flagged as Threat	Alert Flagged as non-threat	No Action Taken
Alert is a threat	True Positive	False Negative	Ignore Threat
Alert is a non-threat	False Positive	True Negative	Ignore Non-Threat

In Binary Classification systems, there are three main calculations done to evaluate the rate of successful classifications, Accuracy, Precision, and Recall (Powers, 2011). Accuracy is the ratio of objects that were classified correctly compared to all objects in the system. Precision is the percent of objects that had a specific trait out of all objects that were said to have the specific trait. Recall is the percent of objects that are correctly said to have a specific trait out of all objects that have the trait. However, the system in this research is different from a Binary Classification system as there is the additional possibility the operator took no action. Therefore, Accuracy, Precision and Recall must be expanded.

For the purposes of this expansion, we use Table 7 to define our variables.

Table 7: Variable Definitions

Variable Name	Definition
TP	True Positives
FP	False Positives
FN	False Negatives
TN	True Negatives
IT	Ignore Threat
INT	Ignore Non-Threat

We now define the following equations to evaluate the success of the classification of alerts.

$$Active\ Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

$$Passive\ Accuracy = \frac{TP + TN}{TP + FP + FN + TN + IT} \quad (2)$$

$$Results\ Accuracy = \frac{TP + TN + INT}{TP + FP + FN + TN + IT + INT} \quad (3)$$

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$Active\ Recall = \frac{TP}{TP + FN} \quad (5)$$

$$Passive\ Recall = \frac{TP}{TP + FN + IT} \quad (6)$$

Accuracy refers to the percent of alerts classified correctly. For this research, Active Accuracy only focuses on the alerts where the operator made a decision, ignoring the alerts where no action was taken. Passive Accuracy also includes those alerts that represented true threats where the operator did not make a decision; however, Passive Accuracy ignores the alerts that were not threats when no action was taken. Results

Accuracy accounts for both types of alerts when no action was taken. Results Accuracy treats the lack of action on an alert that is not a threat as a correct choice because the alert was not flagged as a threat, and is therefore correct, despite the fact that the alert being classified correctly was not due to the operator's decision.

Precision is the only of these calculations that can be taken directly from its normal definition, as it is still the percent of true threats flagged correctly out of all alerts flagged.

Similarly to Active Accuracy, Active Recall only takes into account alerts the operator took action on, while Passive Recall expands Active Recall to the alerts that were true threats but had no action taken on them.

For the purposes of the experiment with IMPRINT, it is expected that Precision, Active Accuracy, and Active Recall will be relatively consistent throughout the changing difficulty of the conditions. This consistency is caused by the fact that these calculations are only based off alerts the operator took action on, and the chance that an alert will be classified correctly if it is classified, is independent of the amount of alerts in the system.

It is expected that as the conditions become harder by the Severity Distribution favoring more severe alerts, and the amount of alerts per minute increase, then the Passive Accuracy, Results Accuracy, and Passive Recall will all decrease due to the alerts increasing to a degree that they surpass the amount the operator can handle. As the conditions become harder, in the beginning, Passive Accuracy, Results Accuracy, and Passive Recall should remain high as the difficulty is not something the operator cannot handle, but there will be a point where the operator won't be able to handle all the alerts, and the values will begin falling. Results Accuracy will still remain relatively high as it will be boosted by the alerts that are not threats that have no action taken on them since these are classified as a correct. Passive Recall is expected to drop the most as there will be nothing to counter balance the higher number of true threats that will be missed as the difficulty increases.

Table 8 summarizes the expected behavior for each performance calculation.

Table 8: Expected Behavior for the IMPRINT Study

Calculation	Estimated change across APM levels	Estimated change across Severity Distribution levels
Active Accuracy	Not significant	Not significant
Passive Accuracy	Significant	Significant
Results Accuracy	Significant	Significant
Precision	Not significant	Not significant
Active Recall	Not significant	Not significant
Passive Recall	Significant	Significant

These hypotheses will be tested by running a two-way ANOVA for each performance calculation, testing the change caused by the APM level, the Severity Distribution level, and the interaction between the two. If the p value from the ANOVA is less than 0.05 that will provide evidence that the changing of the independent factors does affect that performance calculation.

IMPRINT also provides an objective measure of workload at each instant throughout the trial. This workload data makes it possible to calculate a time-weighted average workload across the entire trial. It is expected that the average workload will continue to increase as the conditions become harder, however there will be a cap in which no matter how much harder the conditions become, the workload will barely change. This cap is caused by the fact that the workload at any given time is a sum of all the sub tasks being performed at that time, which leads to the highest workload being while the operator is investigating an alert while also monitoring for severe alerts. The highest average workload will be when the operator spends the most amount of time investigating alerts. Once the operator reaches the limit of how many alerts can be processed, increasing the amount of alerts coming into the system can't raise the amount of time the operator spends investigating the alerts further.

By taking the average workload across the condition and the performance scores, it's possible to create a workload performance profile that shows the relationship between workload and performance (Mitchell, 2000). A workload performance profile typically has 3 main regions of interest. The first main region, the underloaded region, is where workload and performance are both low. This behavior is caused by the task being so boring and lacking stimulus that the person loses interest and performs badly. The next region of interest, peak performance region, is where workload is medium, and performance is high. This behavior is caused by the task being stimulating enough that the human is able to stay engaged, but workload isn't high enough that the human is overwhelmed. This region also represents optimal workload, where the person is comfortable and able to manage the task. The final region, the overloaded region is where the workload is high and the human is overworked, so performance begins to degrade. A sample Workload Performance Profile is shown in Figure 3 below.

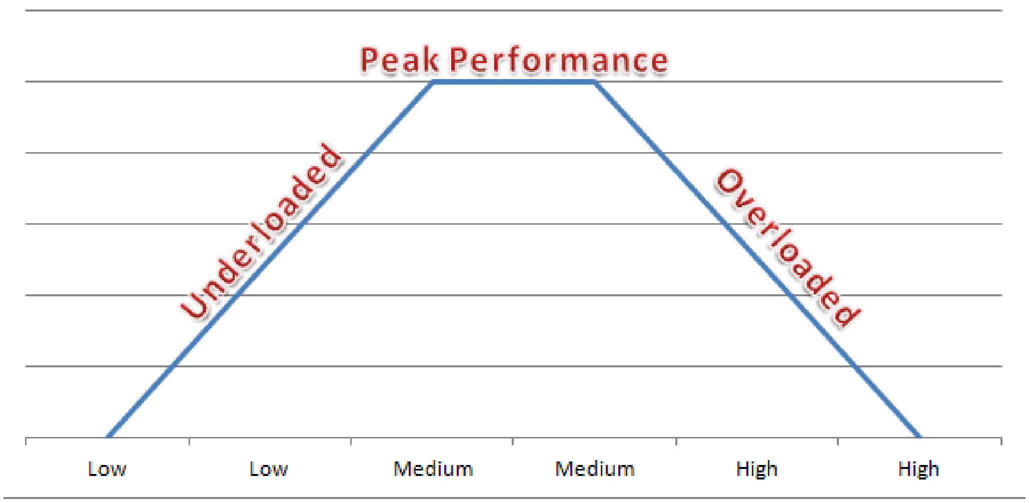


Figure 3: Sample Workload Performance Profile

The major goal of the IMPRINT experiment is to downselect conditions from the 210 conditions being evaluated. These down selected conditions will then be explored using the STE in the STE Calibration Study. The reason downselecting is important is that the IMPRINT model is able to explore many different conditions quickly, while it is much more resource intensive to explore conditions with real people. If an IMPRINT model can be used to identify the levels of factors that create conditions of interest, then it is possible to know that when human subjects are used, the right types of conditions are explored. The method for choosing the 4 conditions to be downselected from the 210 can vary depending on the research question being asked. In some research, the conditions right when the subject is expected to start making mistakes may be a condition of interest, while in other research a condition where the subject makes lots of mistakes may be desired.

For this research, 4 conditions are downselected, and examining where the conditions fall on the Workload Performance Profile is one way to narrow down the conditions of interest. It is unlikely that there will be any conditions in the underloaded region first section, as the task does not last long enough for the human to struggle to stay attentive. Conditions in the peak performance region are those in which the human is not overloaded and able to keep up with the demands of the task. As the goal of the calibration study is to see if the IMPRINT model is correct by neither overestimating nor underestimating the human's performance, conditions in the peak performance region are not interesting, as it's impossible to see if the human performed better than expected if the human is expected to perform at the maximum. Conditions in the overloaded region are the most interesting as this section is where the human begins to be overloaded and cannot handle the task. By choosing the conditions in this range, it's possible to see if the IMPRINT model either underestimated or overestimated the human's performance.

Since there are likely a large number of conditions in overloaded region of the workload performance profile, other factors can be used to narrow down the amount of conditions. The factors that are used depend on the questions the research is asking. For this research, one of the questions that can be explored is at what points does increasing the APM level offset decreasing the Severity Distribution level. This research question could explore how well the IMPRINT model accounts for the changing independent variables. In order to find points where this offset happens, it's necessary to find conditions that have a similar performance calculation, but differing APM and Severity Distribution levels. Specifically, the performance calculations that should be looked at are those in which the ANOVAs indicate a significant difference across both APM and

Severity as indicated by the p values from the ANOVAs, and there is a large range in values for the calculations. The large range in performance values from the IMPRINT study makes it easier to identify a difference in performance during the STE Calibration Study.

In order to downselect the 4 conditions, two pairs of conditions can be selected, with one pair from an easier part of the overloaded section, and the other pair representing a more difficult part of the overloaded section.

Section 3.6: Calibration Study

This section provides an overview of the Calibration Study done with the STE described in section 3.1. This overview includes a discussion on the goals for the experiment, along with an outline of the procedure.

Study Objective

The primary goals of the study are to verify the capabilities of the STE to drive different levels of workload, to collect data correctly, and to demonstrate the capability to synchronize the IMPRINT model's expected workload and performance profiles with the human subject self-reported workload and the performance profiles from the STE, respectively.

Experimental Design: Factors

For the purposes of this study, 2 factors are manipulated. The first is the number of alerts that arrive per minute, and the second is the distribution of alerts that fall into each of the 5 types of severity. There will be a total of 4 conditions downselected from the 210 examined in the IMPRINT study.

All other factors will be kept as constant as possible. One of these factors includes the length of the trial, kept consistent at 10 minutes. The way the alerts enter the system is also kept constant, at the beginning of each minute. Another factor kept constant is the probability that the non-threats should be rejected for a certain reason. For example, the probability that an alert should be identified as a non-threat based on the pcap not fully matching the signature is 75%, a factor consistent with estimates provided by SMEs (Lovell, 2014). The rest of the non-threats are designed such that they are split evenly between three other categories. The first category is benign alerts, those that are not a threat because they represent normal activity. The second category is alerts where the malicious action was already taken care of by the system, such as having a patch against that threat already implemented. The third category is alerts that are not a threat due to the originating IP Address.

Subjects

As this is a proof of concept experiment, only 3 subjects are used. Subjects are trained via a short PowerPoint presentation, and run through a practice trial where the participant is able to ask the researcher questions to ensure they understand the task. After the subjects complete the data collection trials, the subject is given the chance to provide feedback about the training which can aid researchers in identifying what areas may need to be clarified or highlighted.

Procedure

For the data collection, each subject completes 4 trials, one for each condition. The order in which these trials are presented is randomized; however, the order is the same for each subject which allows learning effects to affect the subjects in a consistent way, as opposed to having one condition easier for one subject than the other due to learning effects. Each trial lasts 10 minutes. Following each trial, the participant is asked to fill out the NASA-TLX for that task which can be found in Appendix B. The NASA-TLX makes it possible to examine how a subjective workload measurement compares to the objective workload computed by IMPRINT. Due to the differing scales of the workload provided by IMPRINT, the comparison cannot be done directly, but the two can be compared by examining the correlation between the two. Once the trials are completed the subjects are interviewed to see what they thought was easy or hard about the task, along with giving them the chance to suggest ways to improve the task environment and the experimental process. These questions can be found in Appendix C.

Section 3.7 STE Calibration Study Evaluation Criteria

The STE Calibration Study's purpose is to evaluate the quality of the IMPRINT model. This evaluation is done in two main ways.

The first evaluation criterion is whether the subject's performance matches the values predicted by the IMPRINT model. This evaluation is done by examining the performance metrics that were identified to be helpful in narrowing the conditions. Each of the conditions in IMPRINT is run 15 times, which makes it possible to construct a confidence interval on the predicted average performance. The performance of the subjects can then be compared to the confidence intervals from the IMPRINT model. As this is a proof of concept study with a small amount of participants, it's infeasible to construct a confidence interval around subject performance, nor use a t-test to compare the two groups, though this technique is a possibility for other studies. Instead, each of the subject’s performance scores can be compared to the confidence interval from IMPRINT. Comparing the performance also includes examining the failure rate for investigating alerts between the IMPRINT model and the subjects.

The second evaluation criterion is whether the workload reported by the subjects matches what is predicted by the IMPRINT model. Due to the different scales that IMPRINT’s VACP workload and the NASA-TLX have, the two groups cannot be directly compared. However, the order of the rankings from easiest to hardest can still be used. In addition, correlation calculations can be done to see if there is a relationship between the types of workload.

Summary

This chapter described the process of using a workload analyzer such as IMPRINT to aid in experimental design. This process included identifying Line Analysts as the specific task domain to investigate, and undergoing the process of understanding the task by interviewing those with experience working with or as network analysts. The problem was examined from a researcher’s point of view, seeing how the task could be abstracted while still keeping the core of the task intact, along with seeing what capabilities the researcher would need from such an environment. The first iteration of the STE is designed and built. An IMPRINT model of the task is constructed to perform a discrete event simulation of the task. A sample experiment is designed to calibrate the IMPRINT model. In the next chapter, we present and discuss the findings from the IMPRINT model, and the results from the STE Calibration Study.

IV. Analysis and Results

Chapter Overview

This chapter presents the data for two experiments performed as a part of this research: the IMPRINT Experiment and the STE Calibration Study. The IMPRINT Experiment uses IMPRINT to simulate 210 different conditions of the model described in section 3.3. During the IMPRINT Experiment, the rate of alerts coming into the system is manipulated, along with the Severity of these alerts. The goal of the IMPRINT Experiment is to identify interesting conditions that can be used for a sample experiment using the STE by examining the relationship between workload and performance of the conditions on a Workload Performance Profile. The STE Calibration Study uses the conditions identified in the IMPRINT Experiment in an experiment using the STE. The goal of the Calibration Study is to examine if the results match the workload and performance that were predicted by the IMPRINT Model.

This chapter is composed of three main sections. The first section presents the conditions, results, and analysis of the IMPRINT Experiment, and the second section presents the conditions, results, and analysis of the Calibration Study.

Section 4.1: IMPRINT Experiment Data

This section is composed into two subsections. The first subsection provides an overview of the independent variables manipulated for the IMPRINT experiment, which summarizes the key points identified in section 3.4. The second subsection provides the data and analysis from the IMPRINT experiment.

Section 4.1.1: Independent Factors

For this study, two factors are manipulated. The first is the number of alerts that arrive per minute, and the second is the distribution of alerts that comprise each of the 5 types of severity. All other factors are kept as consistent as possible.

There are 10 levels of the rate of alerts. These range from 1 to 10 alerts per minute. Consistent with a real world environment (Lovell, 2014), all the alerts for the minute will enter the system at the start of the minute.

There are 21 levels for the distribution of the severity of alerts. Table 9 below shows the percent of each severity of alert for each of the severity levels.

Table 9: Severity Level Distributions

Severity Level	Percent Very Low	Percent Low	Percent Medium	Percent High	Percent Very High
1	35	35	20	10	0
2	34	34	20	11	1
3	33	33	20	12	2
4	32	32	20	13	3
5	31	31	20	14	4
6	30	30	20	15	5
7	29	29	20	16	6
8	28	28	20	17	7
9	27	27	20	18	8
10	26	26	20	19	9
11	25	25	20	20	10
12	24	24	20	21	11
13	23	23	20	22	12
14	22	22	20	23	13
15	21	21	20	24	14
16	20	20	20	25	15
17	19	19	20	26	16
18	18	18	20	27	17
19	17	17	20	28	18
20	16	16	20	29	19
21	15	15	20	30	20

A full factorial design is done with these two factors for a total of 210 conditions. Each condition is repeated 15 times, each with a different random number seed. By repeating the conditions with different random number seeds, the chance that a rare event would improperly skew the results is reduced.

For each run of IMPRINT, IMPRINT produces a workload summary, which indicates the total objective workload at each point the workload changes. This workload summary makes it possible to calculate time-weighted average workload across each run. IMPRINT also produces a snapshot of key variables at each second, such as the amount of each category of alerts. These variables include a count for true positive, true negatives, false positives, false negatives, along with information about the alerts not yet worked on. This snapshot makes it possible to calculate how well the simulated operator performed. A Workload Performance Profile can be created comparing the time-weighted average workload and performance, which makes it possible to see if there's a correlation between workload and performance. With the workload performance profile, it is then possible to identify conditions of interest that can be used in an experiment with the STE.

Section 4.1.2: Results and Analysis

This subsection discusses the data related to Workload, and then performance for the IMPRINT experiment.

IMPRINT provides workload values for seven channels of workload at every point in the trial that the values change. Due to the nature of the task, only the Visual, Cognitive, and Fine Motor are examined for this task. The workload at each point of the trial is the sum of the values for these three channels of workload. The workload for the trial is found by calculating the time-weighted average of all the points of workload.

Figure 4 illustrates the average workload values from the IMPRINT model for each level of Alerts Per Minute (APM) examined, and Figure 5 illustrates the average workload values across each of the Severity levels. As predicted in chapter 3, the workload increases as APM increases, with a leveling off point where the workload no longer increases as APM increases. A two-way between conditions ANOVA was conducted to compare the effects of APM and Severity levels on Workload. There was a significant effect of the APM level on Workload at the $p<0.05$ level [F (9, 3234) = 8372, $p=0$]. There was a significant effect of the Severity level on Workload at the $p<0.05$ level [F (20, 3234) = 132.7, $p=0$]. There was also a significant interaction between APM and Severity on Workload [F (180, 3234) =6.47, $p=0$]. Running a multiple comparison test of workload between the APM levels indicates that the APM levels of 4 through 10 have no significant difference of average workload between them, while 1, 2, and 3 Alerts Per Minute are significantly different from all other levels, and from each other. This comparison demonstrates that the leveling off point for workload occurs at around 4 Alerts per Minute. The reason workload does not continue to increase much at this point is due to the fact that the IMPRINT model is measuring the workload of the task that the operator is doing, and the model does not increase workload just by having alerts piling up. Adding more alerts while the operator is already working at capacity will not increase workload because the operator does not take any action on the alerts.

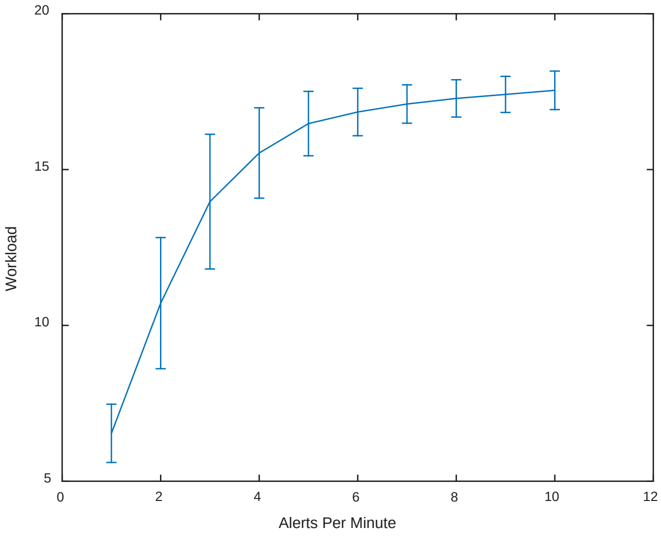


Figure 4: Workload across Alert per Minute Levels

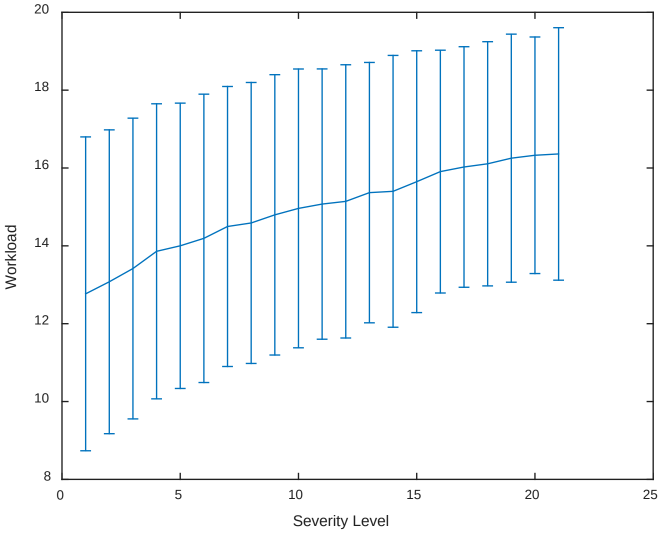


Figure 5: Average Workload Across Severity Levels

Figure 6 provides a 3 dimensional graph of Workload against both Alerts per Minute and Severity. The graph indicates that for low levels of APM, increasing severity tends to increase workload. As indicated by the previous graphs, workload approaches a ceiling as the task difficulty increases. There is still some increase even after this ceiling; however the increase is much smaller than before the difficulty reaches the ceiling.

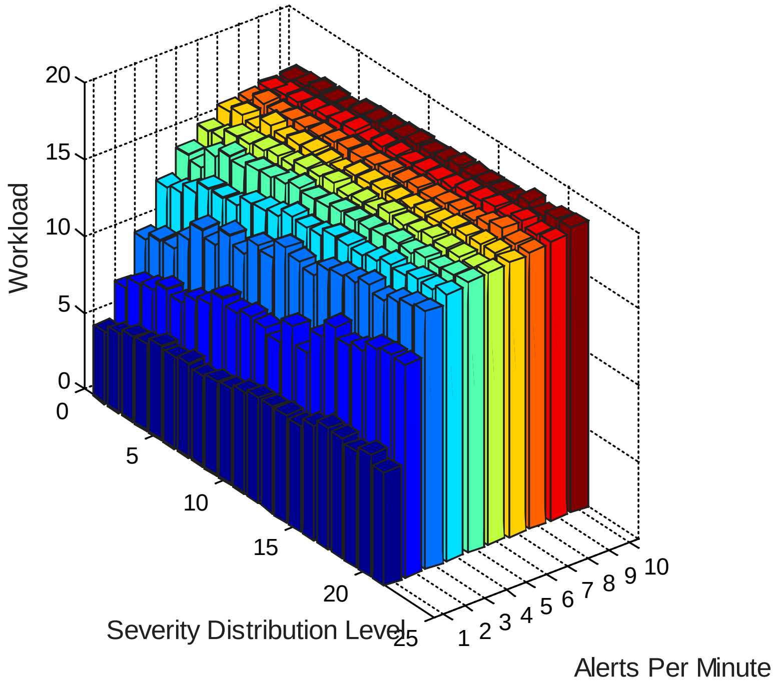


Figure 6: Workload Across APM and Severity Levels

Figure 7 presents the average Active Accuracy across all levels of APM, and Figure 8 presents the average Active Accuracy across all Severity levels. As expected, the values remain high most of the time, though a visual inspection could indicate a downward trend as APM increases. A two-way between conditions ANOVA was conducted to compare the effect of APM and Severity levels on Active Accuracy. There was a significant effect of APM on Active Accuracy at the p<0.05 level [F (9, 2940) = 4.87, p=0]. There not a significant effect of the Severity level on Active Accuracy [[F (20, 2940) = 1.4, p=0.112]. There was also not a significant effect of an interaction between APM and Severity levels on Active Accuracy [[F (180, 2940) = 0.88, p=0.86].

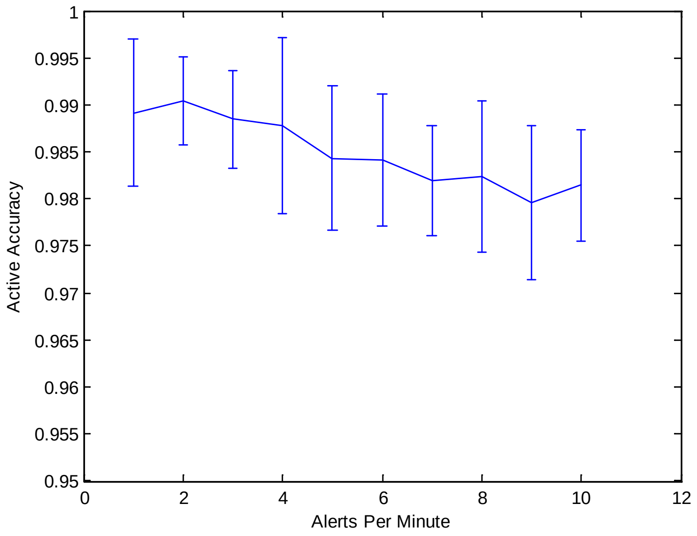


Figure 7: Active Accuracy Across APM Levels

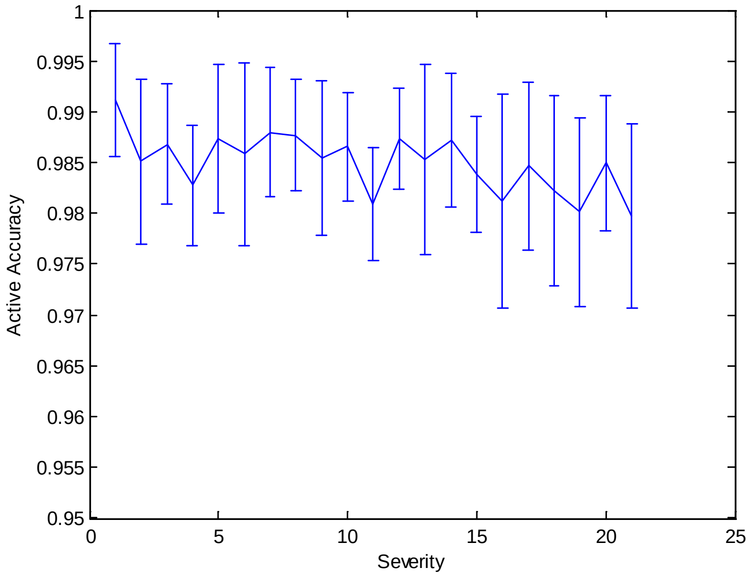


Figure 8: Active Accuracy Across Severity Levels

There being a significant change for Active Accuracy across APM levels is unexpected, as it was hypothesized that Active Accuracy should remain relatively constant across APM levels. After further investigating the IMPRINT model, a possible discrepancy is noted where due to the different amount of steps that an alert can take to investigate, the chance of failure is higher for some alerts than others. Since alerts that are true threats always require the maximum amount of checks performed, having a larger amount of true threats could result in a lower accuracy. True threats taking longer to investigate is also true in the real world as the alerts can’t just be discarded right away when the analyst sees that it’s benign traffic or doesn’t match the signature. Since as severity increases, the amount of true threats also increases, a difference of accuracy caused by changing severity is a possibility, however there was not a significant difference caused by severity. However, that still doesn’t explain the difference caused by the different APM levels. This unexpected behavior could be explained by the increase in the total number of alerts resulting in an increase in the amount of severe alerts. The operator in the IMPRINT model investigates the higher severity alerts first, so as APM increases the operator investigates more severe alerts—even when severity is held constant. Since the more severe alerts are more likely to be true threats, the higher APM level, the more likely the operator is investigating true threats. As previously explained, true threats are more likely to be evaluated incorrectly, explaining the difference in Active Accuracy as APM changes.

Figure 9 illustrates the average Passive Accuracy for each of the APM levels, and Figure 10 illustrates the average Passive Accuracy for each of the Severity Levels. As predicted in chapter 3, the Passive Accuracy trends downward as APM increases in what appears to be a linear fashion. By running a linear regression model, it’s possible to obtain a slope and y intercept that fits the data, which has an R^2 value of 0.97 indicating that a linear model closely resembles the data. This matches the expected results, as the main force behind the changing Passive Accuracy levels is a growing number of alerts that cannot be handled, which increases linearly as APM increases. A two-way between conditions ANOVA was conducted to compare the effects of the APM and Severity levels on Passive Accuracy. There was a significant difference of the APM level [F(9,2940)=219, p=0], Severity level [F=(20,2940)=32.81, p=0], and an interaction between the two [F(180,2940)=2.26, p=0] on Passive Accuracy at the p<0.05 level. This difference matches the predictions in chapter 3. The difference caused by the Severity level is less pronounced than APM level, but indicates that as there are more severe alerts, the operator becomes overwhelmed. An interesting note is that there is a large variance at higher APM levels, possibly tying into the strong interaction between the two factors.

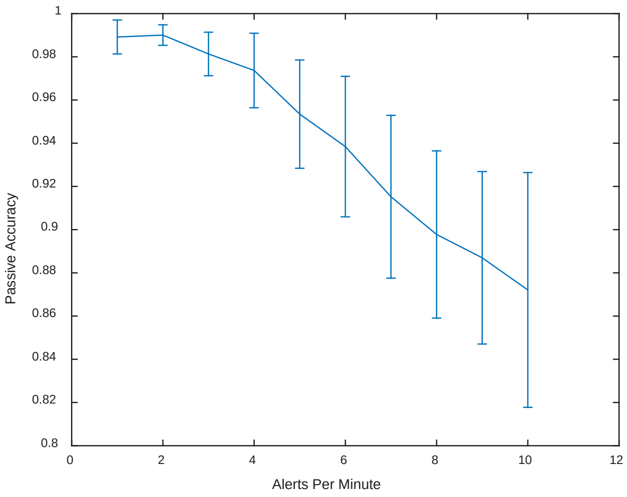


Figure 9: Passive Accuracy Across APM Levels

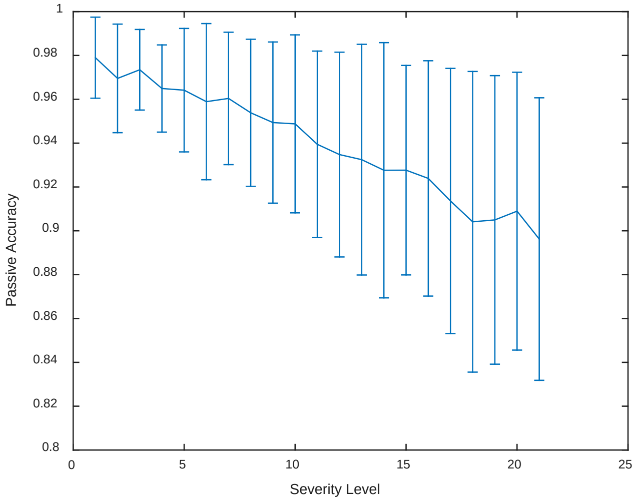


Figure 10: Passive Accuracy Across Severity Levels

Figure 11 illustrates Results Accuracy across the APM levels, and Figure 12 illustrates Results Accuracy across the Severity levels. As predicted in chapter 3, Results Accuracy stays high throughout all APM levels. A two-way between conditions ANOVA was conducted to compare the effects of the APM level, and Severity level on Results Accuracy for each condition. There was a significant effect of APM [F(9,2940)=30.4, p=0], Severity [F(20,2940)=21.09, p=0], and an interaction between the two [F(200,2940)=1.3, p=0.006] on Results Accuracy This difference matches the prediction in chapter 3 that there would be a significant difference across the conditions.

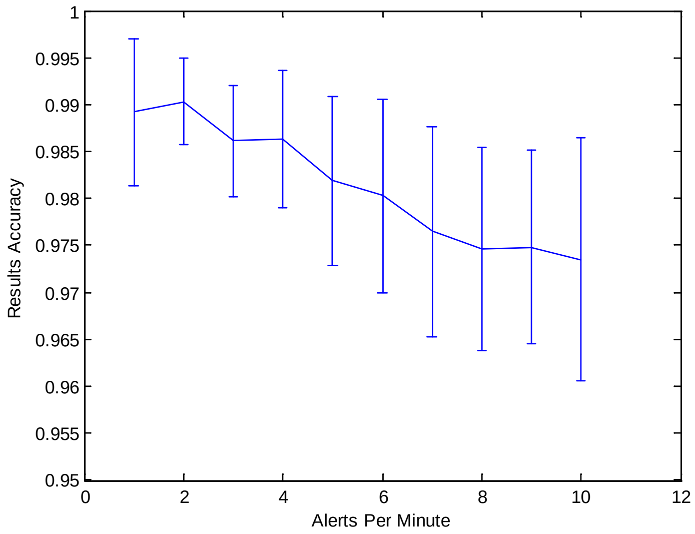


Figure 11: Results Accuracy Across APM Levels

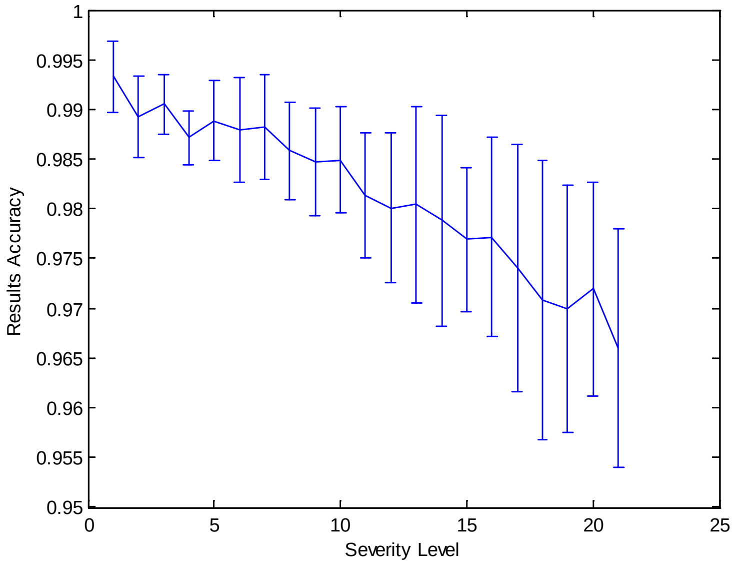


Figure 12: Results Accuracy Across Severity Levels

Figure 13 illustrates Precision across APM levels, and Figure 14 illustrates Precision across Severity levels. Unlike Accuracy, there does not appear to be a relation where Precision decreases as APM increases. Instead, Precision seems to fluctuate between increasing and decreasing across the APM levels, however the large error bars make obtaining information from visual inspection difficult. A two-way between conditions ANOVA was conducted to compare the effects of the APM and Severity levels on Precision. There was not a significant effect of either APM [F (9, 2185[3]) =0.92, p=0.50] nor Severity levels [F (20, 2185) =0.98, p=0.48] on Precision. There was also not a significant interaction between APM and Severity levels on Precision [F (180, 2185) =1.02, p=0.41]. The lack of significant difference of precision across conditions matches the predications made in chapter 3.These high p values indicate that the fluctuations in Precision across conditions is not something that can be attributed to the changing APM and Severity Levels, so Precision should not be used as a guide to finding conditions of interest for the STE Calibration Study.

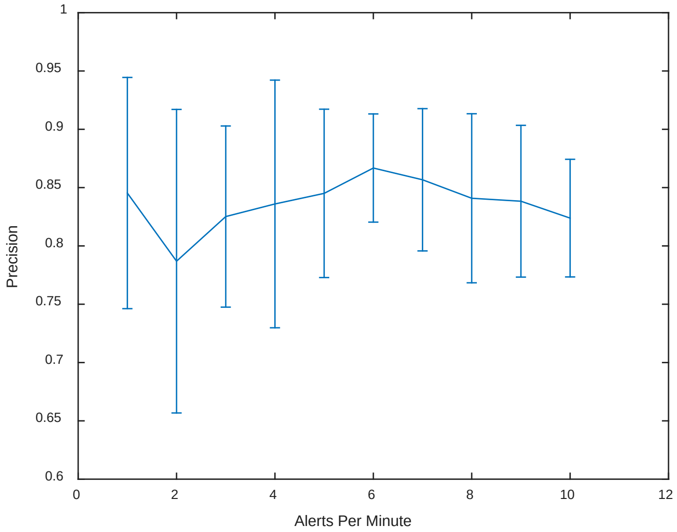


Figure 13: Precision Across APM Levels

[3] Some trials did not have any alerts flagged as threats due to the low frequency of true threats for some conditions, which lowered the amount of samples used for Precision, Active Recall, and Passive Recall.

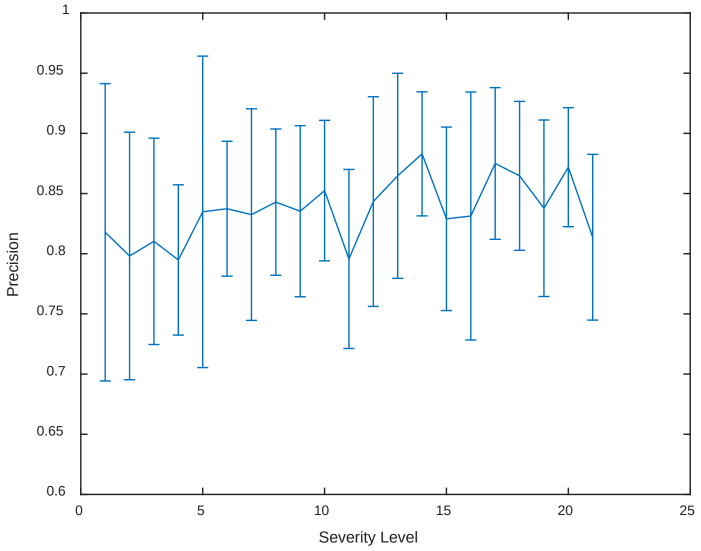


Figure 14: Precision Across Severity Levels

Figure 15 shows a graph of Active Recall across APM levels, and Figure 16 shows a graph of Active Recall across Severity levels. The graph looks similar to that of Active Accuracy with both Active Recall and Active Accuracy slowly decreasing as APM increased. A two-way between conditions ANOVA was conducted to compare the effects of the APM and Severity levels on Active Recall. There was not a significant effect of the APM level on Active Recall at the P<0.05 level [F (9, 2040) =1.59, p=0.11]. There was also not a significant effect of the Severity level on Active Recall at the P<0.05 level [F (20, 2040) =0.59, p=0.59], nor was there a significant interaction between the APM and Severity levels [F (180, 2040) = 1.03, p=0.39]. This result does match the predictions made in chapter 3 where it was expected that Active Recall would remain relatively constant throughout the conditions. Since Active Recall does not vary across both APM and Severity levels, Active Recall is not a good factor to consider when deciding on conditions for the calibration study.

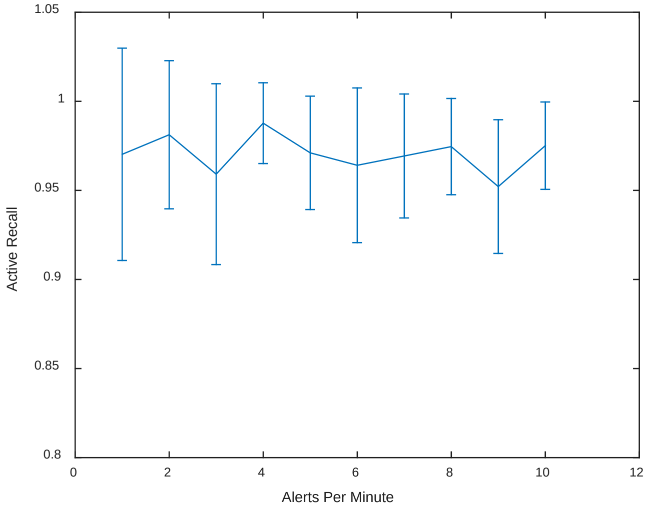


Figure 15: Active Recall Across APM Levels

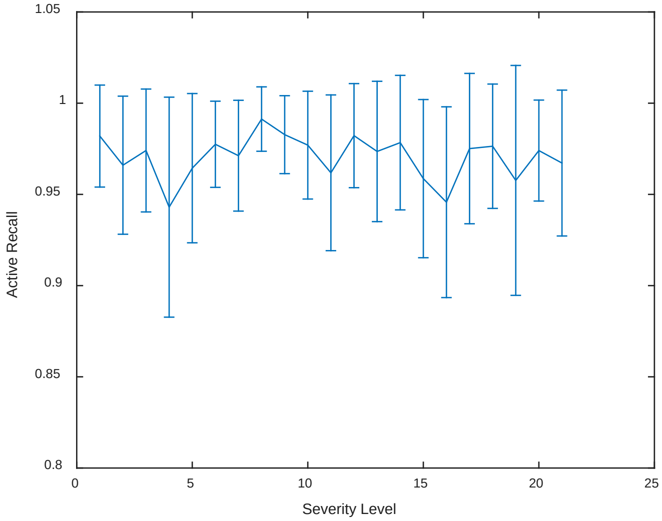


Figure 16: Active Recall Across Severity Levels

Figure 17 shows a graph of Passive Recall across APM levels, and Figure 18 shows a graph of Passive Recall across Severity levels. One of the most noticeable things about these graphs is the scale of the values, with Passive Recall ranging from 0.22 to 1.0 for a condition. A two-way between conditions ANOVA was conducted to compare the effect of the APM and Severity levels on Passive Recall. There was a significant effect of the APM level on Passive Recall at the p<0.05 level [F (9, 2304) =107.51, p=0]. There was also a significant effect of the Severity level on Passive Recall at the p<0.05 level [F (20, 2304) = 9.74, p=0]. However, there was not a significant interaction between APM and Severity levels [F (180, 2304) =0.96, p=0.62]. Considering the significant difference of Passive Recall caused by both the APM and Severity levels, and the large range of values for Passive Recall across the conditions, Passive Recall is a good metric to use to identify the conditions that are downselected for the STE calibration study.

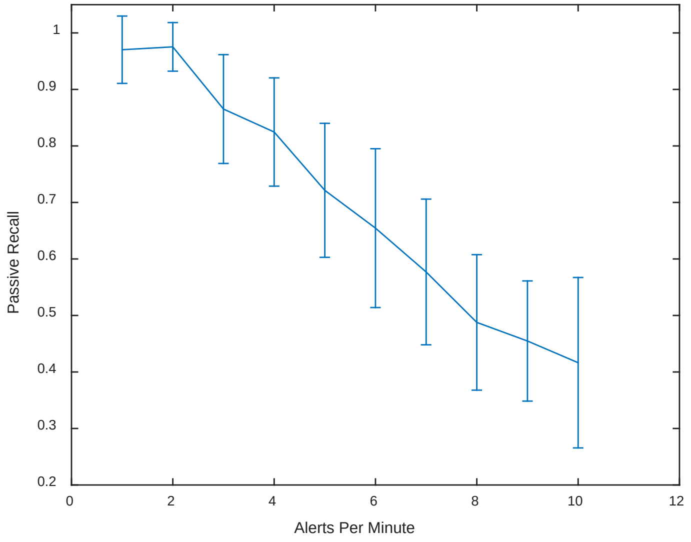


Figure 17: Passive Recall Across APM Levels

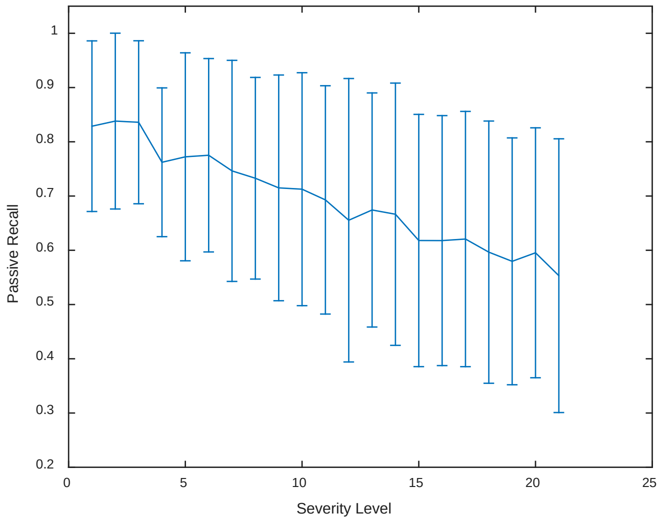


Figure 18: Passive Recall Across Severity Levels

Many of the values explored differed from the predictions made in chapter 3 Table 10 below compares how the performance metrics turned out compared to the predictions in chapter 3. The estimated behavior column indicates the predicted behavior for both changing APM and Severity levels. For all the metrics where there was a significant difference across factors, the values for the metric decreased as APM and Severity increased.

Table 10: Comparison of Predicted to Actual Results

Metric	Estimated Behavior	Effect from APM	Effect from Severity
Active Accuracy	Not Significant	Significant (p=0)	Not Significant (p=0.11)
Passive Accuracy	Significant	Significant (p=0)	Significant (p=0)
Results Accuracy	Significant	Significant (p=0)	Significant (p=0)
Precision	Not Significant	Not Significant (p=0.50)	Not Significant (p=0.48)
Active Recall	Not Significant	Significant (p=0.11)	Not Significant (p=0.59)
Passive Recall	Significant	Significant (p=0)	Significant (p=0)

With many of the calculations shown in this chapter varying only by a few percent from the easiest to the hardest conditions, these calculations are less likely to have the range to be able to match subject performance. Passive Recall has the largest range, with values ranging from 0.22 to 1 for the conditions. The large range makes Passive Recall ideal as a metric to design the STE Calibration Study, as a variation could be identified with a small amount of data from the subjects in the STE calibration study. As one of the questions being explored with the IMPRINT model is the relationship between workload and performance, it’s important to look at how workload and Passive Recall relate to each other. Figure 19 shows a graph of Passive Recall plotted against Workload for all 210 conditions, with the red circles illustrating the areas of interest examined for conditions to use in the Calibration Study. Passive Recall remains relatively high as Workload increases for the most part until around the time that VACP reaches 15, where Passive Recall begins falling. This behavior illustrates that the IMPRINT operator is able to handle the alerts for the most part until workload reaches 15, at which point the operator is unable to keep up.

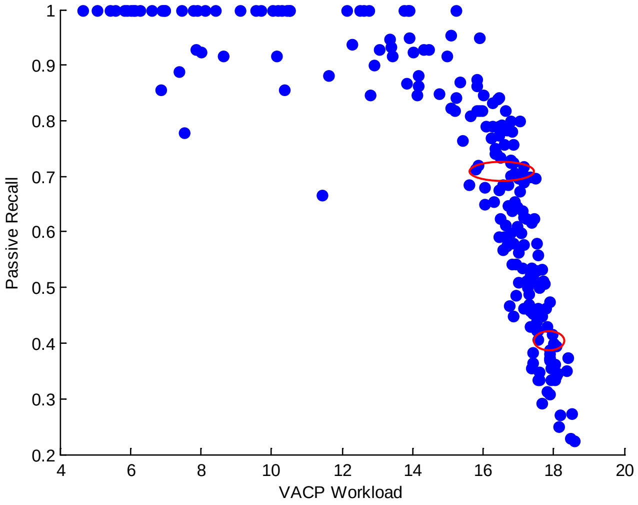


Figure 19: Workload Performance Profile

For the Proof of Concept STE Calibration Study, 4 conditions are to be chosen. One of the factors that lead to a condition being interesting is its place on the workload performance profile. For the purposes of this research, the interesting conditions are those where performance is seen to decrease based on the higher workload.

Due to the fact that there are a large amount of conditions that occur on the curve, to further narrow the possible conditions other factors can be considered. From chapter 3, another factor that makes conditions interesting is those that are similar in performance and workload, but have differing values for the APM and Severity Distribution levels. One condition of the pair should have a higher APM level and the other should have a higher Severity level. By exploring these types of conditions, it’s possible to identify if the IMPRINT model is accurate in taking into account how increasing APM and decreasing Severity affect the operator.

One set of conditions that closely match each other is one where APM=9 and Severity=14, and a condition where APM=8, and Severity =20. These conditions have a less than 1 percentage point difference in Passive Recall (0.406 and 0.400 respectively). They also have very similar average workload (17.6 and 18 respectively). These conditions are further along in the curve so are expected to be harder, though the subjects are predicted to still be able to correctly flag 40% of the true threats.

Another set of conditions that closely match each other is where one condition has APM=6 and Severity=10 and a condition where APM=4 and Severity=18. These conditions are also very close with regards to Passive Recall (0.719 and 0.724 respectively) and Workload (16.9 and 17.1 respectively). These conditions are on the easier side of the curve with a majority of the true threats flagged correctly, however over a fourth of the true threats are undetected.

By choosing conditions on the curve of decreasing performance, it’s possible to check if the IMPRINT model either overestimates or underestimates the amount of true threats the subjects will flag correctly. If the human operator catches all the alerts for the easier conditions, than it can be inferred that the estimates of the IMPRINT simulated operator’s speed should be increased, or the path logic needs changed while if the human operator catches only a quarter of the true threats in the harder conditions, then it can be inferred that the IMPRINT model is incorrect, and that either operator’s speed should be decreased, the failure rate when investigating alerts needs changed, or the path logic needs to be changed.

Section 4.2: STE Calibration Study

This section presents the data and analysis from the Calibration Study done with the STE. This section is composed of two subsections. The first section provides a summary of the factors being manipulated. The second section presents the data and analysis.

Section 4.2.1: Calibration Study Description

For this experiment, 3 subjects are used, each one performing the same 4 conditions specified in Section 4.1.3. The conditions are henceforth named as follows.

- Condition A has APM=9 and Severity=14
- Condition B has APM=4 and Severity=18
- Condition C has APM=6 and Severity=10
- Condition D has APM=8 and Severity=20.

Each of the subjects had some knowledge of the Line Analysts task, with knowledge of the types of the alerts varying from subject to subject.

Before the experiment began, the subjects were led through a PowerPoint presentation which presented instructions for the task. The subjects performed a training condition to become familiar with the STE and the decisions they would need to make. The training condition had 6 alerts spread across the severity levels, and the subjects were not given a time limit to classify the alerts.

The conditions were presented to the subject in numerical order. Each condition lasted 10 minutes. After each condition the subjects were asked to fill out the NASA TLX to assess their subjective workload.

Section 4.2.2: Calibration Study Results and Analysis

Tables 11-13 show the Accuracy, Precision, and Recall calculations for each subject and Table 14 shows the average value from the conditions for the IMPRINT model.

Table 11: Subject 1 Performance Results

	Active Accuracy	Passive Accuracy	Results Accuracy	Precision	Active Recall	Passive Recall
Condition A	1.00	0.96	0.98	1.00	1.00	0.67
Condition B	1.00	1.00	1.00	1.00	1.00	1.00
Condition C	1.00	1.00	1.00	1.00	1.00	1.00
Condition D	0.96	0.96	0.98	0.83	0.83	0.83

Table 12: Subject 2 Performance Results

	Active Accuracy	Passive Accuracy	Results Accuracy	Precision	Active Recall	Passive Recall
Condition A	0.85	0.76	0.92	0.43	1.00	0.50
Condition B	0.95	0.95	0.95	0.60	1.00	1.00
Condition C	0.95	0.95	0.95	0.50	0.67	0.67
Condition D	0.90	0.90	0.91	0.45	0.83	0.83

Table 13: Subject 3 Performance Results

	Active Accuracy	Passive Accuracy	Results Accuracy	Precision	Active Recall	Passive Recall
Condition A	0.74	0.68	0.91	0.40	1.00	0.67
Condition B	0.88	0.88	0.93	0.50	1.00	1.00
Condition C	0.86	0.83	0.92	0.33	1.00	0.67
Condition D	0.86	0.80	0.93	0.50	1.00	0.67

Table 14: IMPRINT Performance Results

	Active Accuracy	Passive Accuracy	Results Accuracy	Precision	Active Recall	Passive Recall
Condition A	0.97	0.87	0.97	0.86	0.89	0.41
Condition B	0.98	0.95	0.98	0.81	1.00	0.72
Condition C	0.99	0.96	0.99	0.96	0.92	0.72
Condition D	0.99	0.86	0.97	0.90	1.00	0.40

The IMPRINT model's estimation of a participant's accuracy in classifying alerts appears to be off. Active Accuracy reflects only those alerts that the subject made a decision on, which for the IMPRINT model in these conditions is always above 97%. The only subject who was overall in this range was subject 1 who had the most knowledge about the types of alerts before participating in the study. This difference means that if someone wants to use IMPRINT to model more novice users, than its rate of failure in investigating the alerts should be increased.

Based on the Passive Recall scores, subjects tended to perform better than the IMPRINT model estimated. Subject 2 stated that in conditions 2, 3, and 4 they were able to catch up on the alerts at times. The IMPRINT model estimated that between a quarter and a half of true threats would remain undetected. This difference indicates that the simulated operator in the IMPRINT model should perform the task faster than is currently programmed.

So far, the values for the IMPRINT model's estimation for Passive Recall have been shown as a single value. This value was calculated by taking the total Passive Recall across the 15 runs performed for the condition. However the single value can be expanded to a confidence interval based on these 15 trials. As described in section 3.4 each of the 210 conditions run with the IMPRINT model was run 15 times, with the only difference between these 15 runs being the random number seed. The random number seed affects everything chosen probabilistically in IMPRINT, such as the time for subtasks that use distributions and if the investigation of a given alert yields the correct answer or not. These confidence intervals are calculated with alpha=0.05 and are shown in Table 15 below.

Table 15: Passive Recall 95% Confidence Intervals for IMPRINT

	Lower Bound	Upper Bound
Condition A	0.312	0.576
Condition B	0.485	0.860
Condition C	0.620	0.898
Condition D	0.280	0.562

For condition A, only subject 2's Passive Recall fell within the bound. For condition B, all subjects had a passive recall score of 1, so do not fall within the bound. For condition C, subject 1 did not fall within the bound, while subject 2 and 3 were within the bound. For condition D, all 3 subjects were higher than the upper bound.

Of the 4 conditions across the 3 participants, only 3 of the 12 Passive Recall calculations fell within the confidence intervals from the IMPRINT model data, with all of the mismatches between the data caused by the subjects scoring higher than the IMPRINT model predicted. These mismatches illustrate that the IMPRINT model is tuned improperly and should be adjusted to better reflect what was seen in the experiments with the STE.

Table 16 provides the Raw TLX scores for each participant, along with the VACP time-weighted average workload estimate from the IMPRINT model. Note that due to the different scales, these two should not be directly compared. It is also important to note that different subjects may have different standards for what they consider low or high workload for the categories so it can be difficult to compare them directly as well.

Table 16: Workload Values for IMPRINT and Subjects

	IMPRINT	Subject 1	Subject 2	Subject 3
Condition A	17.6	27.5	70.00	67.50
Condition B	16.9	15.83	35.83	53.33
Condition C	17.1	14.17	54.17	54.17
Condition D	18	26.67	73.33	64.17

Instead of looking at the numerical values directly, it’s possible to compare how the IMPRINT model and the subjects ranked the trials from easiest to hardest based off the workload scores. These ranks are shown in Table 17 below with 1 indicating the easiest and 4 indicating the hardest.

Table 17: Workload Ranks for IMPRINT and Subjects

	IMPRINT	Subject 1	Subject 2	Subject 3
Condition A	3	4	3	4
Condition B	1	2	1	1
Condition C	2	1	2	2
Condition D	4	3	4	3

All the subjects and the IMPRINT model agree that Conditions A and D are the harder conditions, with Conditions B and C being the easier conditions. Both subjects 2 and 3 stated in post experiment interviews that Conditions A and D felt about the same, while Conditions B and C also felt about the same in terms of difficulty. These rankings indicate that the subjects believe APM affects their workload more than severity.

There are three changes that can be made in the IMPRINT model to allow the model to better fit behavior seen from the Calibration Study. The first change is to adjust the nodes within the task network diagram. Modifying the nodes could include adding or removing certain nodes, or changing the order of the nodes. While the IMPRINT model is made based off the STE, and the expected behavior of the subjects while using the STE, the subjects did not always match expected behavior. For example, the subjects were expected to monitor for new severe alerts showing up, which not all subjects did, so an example of modifying the nodes would be removing the nodes that included monitoring for severe alerts while investigating another alert if the participant did not continuously monitor for new severe alerts. An example of changing the order of nodes would be during the investigation process, if the subjects investigated the parts of the alert in a different order than the subject was predicted to do. A second change is to adjust the failure rate at each node. For example, during the Calibration Study, participant 2 was accurate 92% of the time when investigating a true threat, so the IMPRINT model can be adjusted to match this value. Currently, there is a 1% failure rate at each step that requires a decision within the process of investigating the alert. The last change is to adjust the amount of time each step takes. During the Calibration Study, the participants investigated alerts faster than the IMPRINT model predicted, so the time each step of the investigation took can be adjusted to better reflect the values seen in the Calibration Study.

Expanding the experiment to acquire more data could also help in pinpointing where values could be adjusted. Possible expansions could include having the same condition run multiple times, or having the conditions last longer than 10 minutes to reduce the impact a single mistake could have on performance. Further simulations with the IMPRINT model could also be run, only focusing on the 4 conditions of interest here, which would allow for more trials to be run instead of just the 15. These additional trials would cut down on the standard deviation allowing for smaller confidence intervals.

Information was also obtained through post experiment questionnaires and observations of the subjects performing the experiment. This information could be used to further tune the IMPRINT model, or improve the functionality of the STE. The participants noted that 10 minutes per trial didn’t seem like too long, and it would be safe to extend time of the trial to a degree without the length of the trial causing them to lose interest. A trial length of 20 minutes was suggested to the subjects; however their responses were divided of that being a reasonable length. One subject commented that they focused on the alert types they recognized first, rationalizing that they could get through them faster so focusing on these was an efficient use of their time. In some cases, the subjects’ behavior was different. One subject focused mainly on the alert they were currently investigating, not acting if a high priority alert came in during the investigation, while the other would drop the current alert if they saw a high priority one. Another difference between subjects was how cautious they were. One subject erred on the side of caution, flagging several alerts they weren’t sure of as a threat, not wanting to miss any true threats. The subject’s strategies could be affected by the training they were provided.

For this experiment, the subjects were told that their tasks, in order of importance were to:

1. Flag All True Threats
2. Minimize flagging non-threats as true threats
3. Flag Non-Threats as Non Threats.

The training could be modified to instruct the participant that the speed in response mattered, or better highlight finding true threats.

Another interesting piece of information from the subjects was their mindset during the experiment. One subject indicated that they treated the task like a game, with catching up on the alerts indicating that he won the game.

V. Conclusions and Recommendations

This thesis has presented research showing how workload analyzers such as IMPRINT can be used to predict subject performance and workload for a task. The overall goal is to have a cyclic interaction between an IMPRINT model and real world experiments, where information acquired from an IMPRINT model can point researchers to interesting conditions to explore, and data from real world experiments can be used to fine tune processes within the IMPRINT model. This thesis has presented the first step in this overall goal, demonstrating how the IMPRINT model can be built, and then used to guide experimental design.

While the IMPRINT model provides a basic simulation based on the task performed with the STE, the model can be improved by analyzing the results of the STE Calibration Study to identify more precise timings for the subtasks, along with the chances of failure at each point. The model can also be refined by updating the chain of decisions to better match the order the subjects made decisions. Model refinement may involve creating multiple models, based on different subjects and the varying techniques they used. These models could vary based on the order the decisions were made, along with the time it took the subjects to make each decision. Varying models could allow different strategies to be compared in IMPRINT, and could possibly lead to research that could identify a best strategy to suggest that analysts could use. The IMPRINT model could also be enhanced to have accuracy and speed fluctuate based on how rushed the operator feels.

The STE designed in chapter 3 could also be expanded to be better reflective of real world environments. This expansion could include making it possible for a team of subjects to work on the task at the same time. More experiments could be done with the STE to explore how a subject performs when there may be connections between alerts. One example of a connection could be where one alert will give the information that's needed to solve an alert that doesn't appear for another 5 minutes.

Further research could also be done to explore how IMPRINT may be applicable to other tasks within cyber, such as forensics, and cyber attack operations. IMPRINT may also have potential in other domains that Human Factors researchers may be interested in such as RPA pilots, though it would be important to consider how differences between domains may affect the required methodology.

IMPRINT itself could also be improved to allow it to better serve the needs of researchers. For this research, IMPRINT version 4.1.278 is used. Some of its flaws are its inability to run multiple trials at once while providing a separate output file for each run, and its somewhat limited programming support, such as its lack of built in object oriented programming. These problems could be addressed through the use of plug-ins, or by building a wrapper around IMPRINT. Another part of IMPRINT that could be improved is its efficiency when multiple runs are performed. Through the experiments, it was found that the RAM IMPRINT occupies increases as the simulations are repeated to the degree that after several hundred runs, IMPRINT takes up about 5 times as much memory as after a single run. In addition to the memory problems, creating the output files after each run becomes progressively slower. While these issues are barely noticeable when the consecutive number of runs is in the 10s, it becomes problematic when hundreds or thousands of consecutive trials are desired to be run.

Another area of future would be to study how tools other than IMPRINT can be used to model the task. For example, Act-R is a modeling architecture that simulates human cognitive processes, though in a different way than IMPRINT (Budiu, 2013). Act-R may be able to be used in addition or in place of IMPRINT to gain a better understanding of the task.

Appendix A: Cyber STE Design Document

Definitions

Subject/Participant: These are people brought in to perform the task. They are assumed to have minimal technical expertise, with all the knowledge about the task provided by a short training session.

Researcher: These are the people who run the experiments. They can be assumed to have some knowledge of the task, enough to train the subjects. They can also be assumed to have some technical abilities (such as modifying config files and parsing output files), but not the ability to modify the STE's code.

Analyst: These are the real world analysts of the 33rd Network Warfare Squadron,

Trial: A run of the STE.

Event: Something suspicious that happens in the real world, or is pretended to happen.

Alert: A record of an event that is displayed in ArcSight/the STE

Purpose

This simulation is to be a Simulated Task Environment (STE) that emulates the environment uses by the Network Analysts of the 33rd Network Warfare Squadron. The STE is to have a similar look and feel to the actual environment, but present information in such a way that it is possible for someone without the technical experience of the analysts to be able to use the STE with minimal training.

The end goal if this STE is to serve as an environment Human Factors researchers can use to study analysts. The intent is for the STE to be as realistic as possible, with any deviations from realism properly justified, especially with the types of choices the subjects need to make, while also being abstract enough so that the subjects can understand and perform the task with minimal training. It also needs to be adequately customizable by the researcher to meet their needs. The STE could be used to examine when an analyst could become overworked, along with future versions possibly providing adaptive automation to alleviate this workload. It could also be used as a training tool, to allow sometime to train in working in the required mindset without needing to worry about all the technical details, though further research may be required to identify the level of abstraction required to provide training benefit. Furthermore, it could also be used to study different interface options, to see how different interfaces would affect the analyst's ability to understand and process the incoming information.

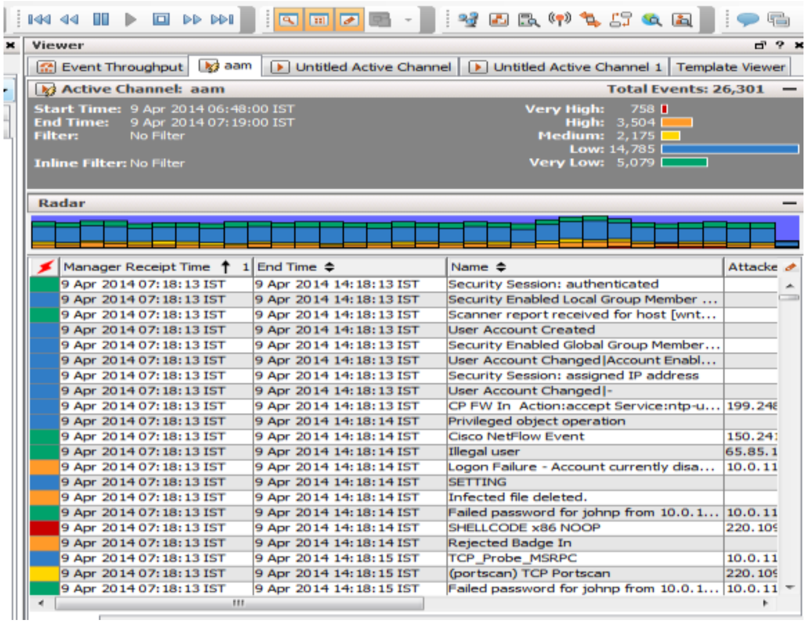
This STE also serves as a case study of how to build an STE in a new domain, and illustrate the process required to make the STE in such a way as to convey realism and meet researcher needs.

Programming Language

The STE will be programmed in C

Real World Task Description

The job of the network analysts is to monitor alerts created by possibly suspicious activity at the gateway of the Air Force's network. A computer program called ArcSight compiles alerts generated by various sensors and displays them for the user. An example screen shot is shown in Figure 20 below. These sensors can include firewalls, intrusion detections systems, email servers, etc. ArcSight displays these alerts in a scrolling window with some basic information. If the analyst believes the alert may indicate a true threat, they will click on it to bring up more information, such as the packet capture (pcap) file. From this information, the analyst will determine if the alert needs to be sent to the Incident Response Team, or can be safely ignored.



Active Channel: aam

Total Events: 26,301

Start Time: 9 Apr 2014 06:48:00 IST
End Time: 9 Apr 2014 07:19:00 IST
Filter: No Filter

Inline Filter: No Filter

Very High: 758
High: 3,504
Medium: 2,175
Low: 14,785
Very Low: 5,079

⚡	Manager Receipt Time ↑ 1	End Time ⇕	Name ⇕	Attacke
🟩	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	Security Session: authenticated	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	Security Enabled Local Group Member ...	
🟩	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	Scanner report received for host [wnt...	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	User Account Created	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	Security Enabled Global Group Member...	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	User Account Changed|Account Enabl...	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	Security Session: assigned IP address	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	User Account Changed|-	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:13 IST	CP FW In Action:accept Service:ntp-u...	199.248
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	Privileged object operation	
🟩	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	Cisco NetFlow Event	150.241
🟩	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	Illegal user	65.85.1
🟧	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	Logon Failure - Account currently disa...	10.0.11
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	SETTING	
🟧	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	Infected file deleted.	
🟩	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	Failed password for johnp from 10.0.1...	10.0.11
🟥	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	SHELLCODE x86 NOOP	220.109
🟧	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:14 IST	Rejected Badge In	
🟦	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:15 IST	TCP_Probe_MSRPC	10.0.11
🟨	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:15 IST	(portscan) TCP Portscan	220.109
🟩	9 Apr 2014 07:18:13 IST	9 Apr 2014 14:18:15 IST	Failed password for johnp from 10.0.1...	10.0.11

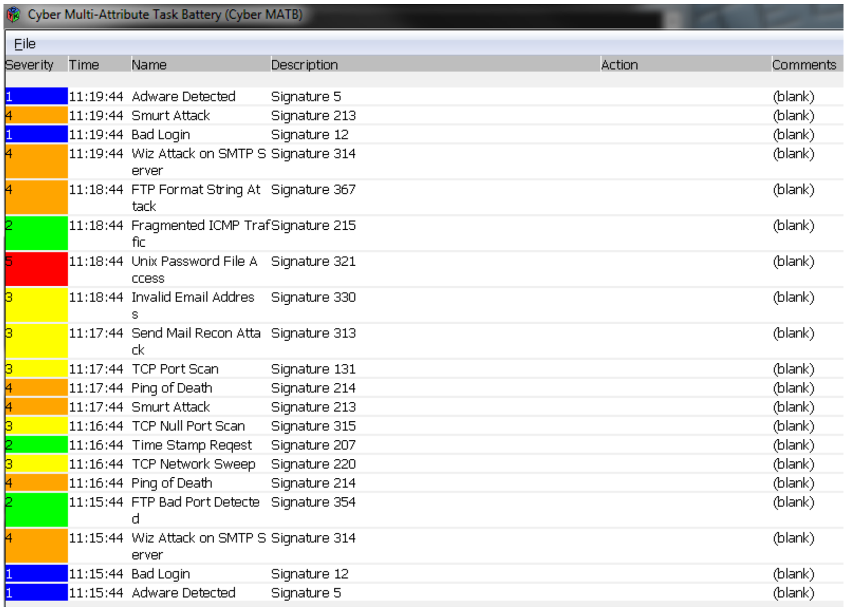
Figure 20: Example ArcSight Window

GUI Requirements

There needs to be a number of different interactions between the user and the STE. For the 33rd NWS, this is shown across two monitors, with ArcSight on one monitor, and the second being used for the other required tools, and at times, internet research. However, this STE is only on one monitor. On the left half is the Alert Window; on the right is the Google/Lookup window, with the communications window at the bottom. The Investigation Windows is popup windows, where the subject can have multiple Investigations up at one time.

Alert Window

This display shows the alerts coming in a scrolling table (similar to Excel), with the timing of when the alerts are displayed specified by a csv file, with several alerts appearing every minute. The STE needs to hold at least 2,000 alerts. The Alert window is shown in Figure 21 below



Severity	Time	Name	Description	Action	Comments
1	11:19:44	Adware Detected	Signature 5		(blank)
4	11:19:44	Smurt Attack	Signature 213		(blank)
1	11:19:44	Bad Login	Signature 12		(blank)
4	11:19:44	Wiz Attack on SMTP Server	Signature 314		(blank)
4	11:18:44	FTP Format String Attack	Signature 367		(blank)
2	11:18:44	Fragmented ICMP Traffic	Signature 215		(blank)
5	11:18:44	Unix Password File Access	Signature 321		(blank)
3	11:18:44	Invalid Email Address	Signature 330		(blank)
3	11:17:44	Send Mail Recon Attack	Signature 313		(blank)
3	11:17:44	TCP Port Scan	Signature 131		(blank)
4	11:17:44	Ping of Death	Signature 214		(blank)
4	11:17:44	Smurt Attack	Signature 213		(blank)
3	11:16:44	TCP Null Port Scan	Signature 315		(blank)
2	11:16:44	Time Stamp Reqest	Signature 207		(blank)
3	11:16:44	TCP Network Sweep	Signature 220		(blank)
4	11:16:44	Ping of Death	Signature 214		(blank)
2	11:15:44	FTP Bad Port Detected	Signature 354		(blank)
4	11:15:44	Wiz Attack on SMTP Server	Signature 314		(blank)
1	11:15:44	Bad Login	Signature 12		(blank)
1	11:15:44	Adware Detected	Signature 5		(blank)

Figure 21: Alert Window Design

The default columns should include, the alert name (i.e. Spam Email), the Alert Description, Severity Level, Analyst Action Taken, and Analyst Comment. The time the event occurred is calculated by a time given for the start of the trail plus the amount of time passed since then for the alert to occur.

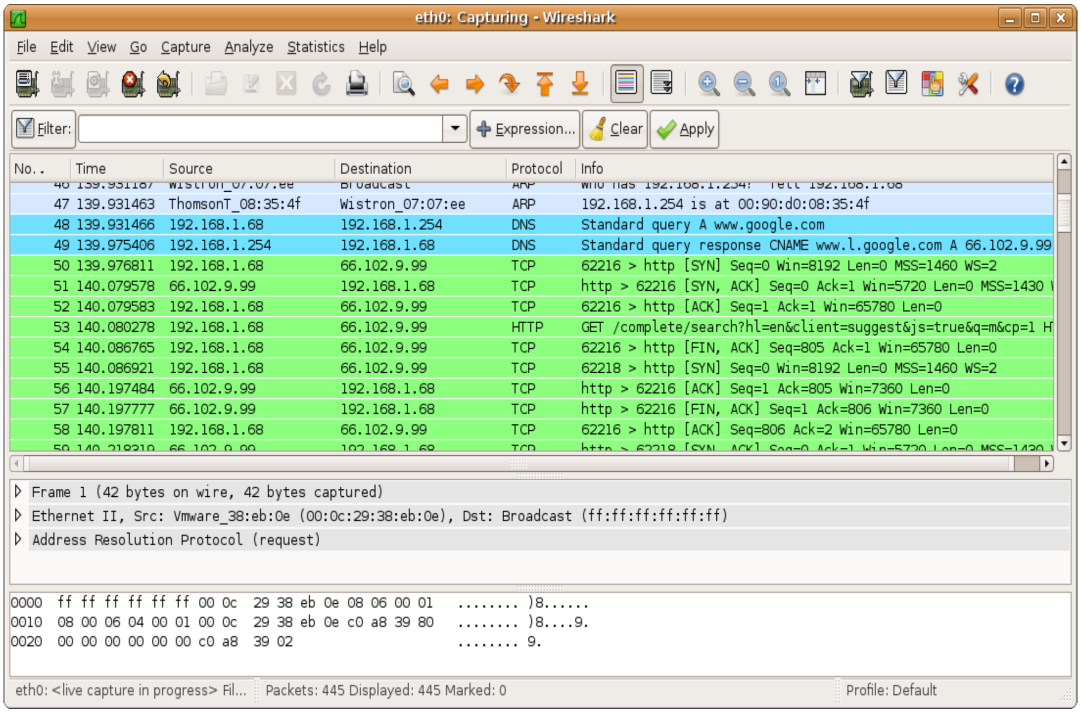
A csv will provide how the numbers representing the severity level in the input file correspond to the colors to be displayed, allowing this to be researcher customizable.

The analyst action taken is populated as the STE runs based on what the subject does. For example, if the subject flags the alert as a threat, this needs to be updated. It is possible for the subject to right click on the Analyst Action Taken button to flag the alert as a non-threat, allowing them to do so without needing to investigate the alert. This does not match ArcSight, however in this case sacrificing realism allows researchers to see a difference between subjects choosing not to investigate because they knew they didn't need to versus the subject not investigating because they didn't notice that alert. This can be configured such that the researcher could turn this capability on and off, so that they could control how realistic they want the STE to be.

For the present version, the Comment section is static, only showing a pre-specified message, however, in later versions this could be expanded to allow the subject to enter notes here, especially if the subject is working in a team. This is often used by the 33rd NWS to indicate who is working on the alert. For example, "Alert being investigated by Greg."

Investigation Window

When the user clicks on an alert in the Alert Window, that alert's Investigation Window comes up. The information displayed includes a pcap file, (similar to what Wireshark displays as seen in Figure 22 below), information about IPs involved, and may include the action the sensor that created the alert took, or other information provided by the sensor. This window has the ability for the subject to make a decision on whether an alert is a true threat or not with buttons.


eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

No. .	Time	Source	Destination	Protocol	Info
46	139.931187	Wistron_07:07:ee	Broadcast	ARP	Who has 192.168.1.254? Tell 192.168.1.68
47	139.931463	ThomsonT_08:35:4f	Wistron_07:07:ee	ARP	192.168.1.254 is at 00:90:d0:08:35:4f
48	139.931466	192.168.1.68	192.168.1.254	DNS	Standard query A www.google.com
49	139.975406	192.168.1.254	192.168.1.68	DNS	Standard query response CNAME www.l.google.com A 66.102.9.99
50	139.976811	192.168.1.68	66.102.9.99	TCP	62216 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
51	140.079578	66.102.9.99	192.168.1.68	TCP	http > 62216 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430
52	140.079583	192.168.1.68	66.102.9.99	TCP	62216 > http [ACK] Seq=1 Ack=1 Win=65780 Len=0
53	140.080278	192.168.1.68	66.102.9.99	HTTP	GET /complete/search?hl=en&client=suggest&js=true&q=m&cp=1 H
54	140.086765	192.168.1.68	66.102.9.99	TCP	62216 > http [FIN, ACK] Seq=805 Ack=1 Win=65780 Len=0
55	140.086921	192.168.1.68	66.102.9.99	TCP	62218 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
56	140.197484	66.102.9.99	192.168.1.68	TCP	http > 62216 [ACK] Seq=1 Ack=805 Win=7360 Len=0
57	140.197777	66.102.9.99	192.168.1.68	TCP	http > 62216 [FIN, ACK] Seq=1 Ack=806 Win=7360 Len=0
58	140.197811	192.168.1.68	66.102.9.99	TCP	62216 > http [ACK] Seq=806 Ack=2 Win=65780 Len=0
59	140.218319	66.102.9.99	192.168.1.68	TCP	http > 62218 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430

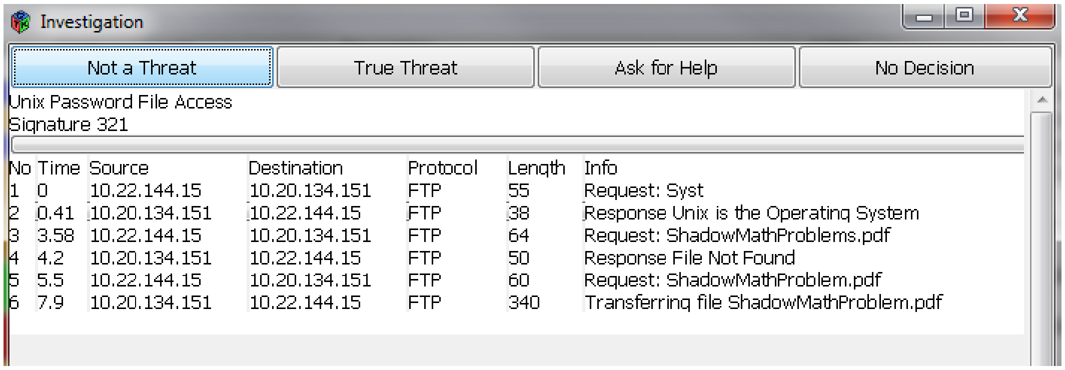
```
▷ Frame 1 (42 bytes on wire, 42 bytes captured)
▷ Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 00 0c  29 38 eb 0e 08 06 00 01   ........ )8......
0010  08 00 06 04 00 01 00 0c  29 38 eb 0e c0 a8 39 80   ........ )8....9.
0020  00 00 00 00 00 00 c0 a8  39 02                     ........ 9.
```

eth0: <live capture in progress> Fil... Packets: 445 Displayed: 445 Marked: 0 Profile: Default

Figure 22: Wireshark Example

Beyond the buttons at the top, the rest of the Investigation window is a series of text boxes and tables, typcially one text box followed by a table, though based on the input file, this can change. See the Input Files section for examples of how a csv would be translated into the data shown in the investigation window. An example of the Investigation window is shown in Figure 23 below.



Unix Password File Access
Signature 321

No	Time	Source	Destination	Protocol	Length	Info
1	0	10.22.144.15	10.20.134.151	FTP	55	Request: Syst
2	0.41	10.20.134.151	10.22.144.15	FTP	38	Response Unix is the Operating System
3	3.58	10.22.144.15	10.20.134.151	FTP	64	Request: ShadowMathProblems.pdf
4	4.2	10.20.134.151	10.22.144.15	FTP	50	Response File Not Found
5	5.5	10.22.144.15	10.20.134.151	FTP	60	Request: ShadowMathProblem.pdf
6	7.9	10.20.134.151	10.22.144.15	FTP	340	Transferring file ShadowMathProblem.pdf

Figure 23: Investigation Window

Description of Real World Alerts

An IDS system (such as CISCO) monitors the network traffic and creates an alert for something it sees as suspicious which will then be sent to ArcSight. These alerts can vary based on the protocol where the suspicious activity occurred, along with the level of detail regarding the event. There may also be information irrelevant to the event in question.

What it looks like in the STE

The alert comes in with the type of alert, along with the signature number. The user can then use the Google/Lookup window to look up the signature. The user can then compare the information in the signature to the information they have to see if the alert matches the signature match, or if the system incorrectly flagged the alert.

One example of an alert that may be a mismatch is a command injection attack caused by inputting an escape character in the input field. The alert could be caused by a malicious user trying to access the data on the system, or by a benign user mistyping. Other examples of alerts could be signatures caused by an attacker trying to get in a back door.

The action the system took could also determine if the alert is a true threat. If the attacker tried once, but the system blocked the attack, then there’s no reason to flag the alert. Conversely, if the attacker tried multiple times, and/or the information showed that data was leaked, the subject should flag the alert. The signature look up could also provide information about the danger caused by something triggering the signature, to guide the subject in determining if the alert was a threat or not. For example, a ping reply can create an alert, but it’s normally nothing to worry about, so the signature should instruct the subject that the alert shouldn’t be flagged as a threat.

Steps to Investigate an Alert

- Step 1, Decide if it needs to be investigated
  - Normally based on severity
  - Other information may also point out a certain threat to always investigate
- Step 2, Find Signature
  - A signature number will be in the information at the top of the investigation window.
  - Also in the information in the Alert Window, under Alert Description
- Step 3, look up signature in Google window
  - This will give the subject an idea of what the alert looks like, a pattern to look for in the pcap
    - Good chance the alert won’t match the signature, if so, then it’s a non-threat
    - May be a small difference, or completely off
  - Signature also indicates if the alert should be flagged assuming the alert matches the signature
    - May be conditional, i.e., if it’s from a malicious IP, or a bad location, it’s a threat, otherwise it’s not
- Step 4, look through pcap to see if it matches the signature.
- Step 5, if it does, and signature pages indicates it’s a threat, then it’s a threat.
  - Otherwise, it can be discarded as a non-threat.

Google/Lookup/Query Window

This window allows the user to query for information. This can include either an IP address or a web URL, and tell the subject if that IP/url is known to be suspicious/malicious or not. It can also be used to query about a signature seen in a file. Combining several tools together such as IP lookup tools and IDS documentation files isn’t entirely realistic, however these tools all serve the same purpose, and use a similar thought process. The researchers would provide an input file which would map keywords to information, with the window displaying information matching any of the words typed in. Depending on the needs of the researchers, this window could also include irrelevant information that the subject would be required to sift through to find what they needed.

Communications Window

This window simulates communication coming in from various sources. These communications occur at a predefined time, with their information specified in the similar way to the alerts coming in. Some of the communications could be irrelevant, targeted toward another team member, or just chatter that the subject shouldn't care about. Some of the communications could reference an alert, or a type of alert, steering the subject to something that's a threat, or guiding them away from something that isn't a threat. One communication may supersede a previous communication as well.

Input Files

Alerts.csv- Holds the information that will be displayed in the alert window, and points to the popup file for that corresponding alert. One Alert file per trial. This provides general information about the alert, that the subject then can use to identify which alerts need to be investigated. An example of this file can be seen in Table 18 below.

Table 18: Example Alert.csv File

Alert Num	Time	Name	Description	Severity	Popup File	Threat	Init Comment	Group
1	5000	Scan	Signature 22	2	Popup1.csv	1	(Blank)	-1
2	7500	Remote Logon	Admin account logged in…	4	Popup2.csv	-1	Worked by Analyst 2	2

The Threat field in the Alert.csv specifies if the alert is a threat or not. This will be a number for the different possibilities. These numbers are

- 0=Not a threat.
- 1=True threat

Other options can be added if a researcher wants to use them.

Colors.csv

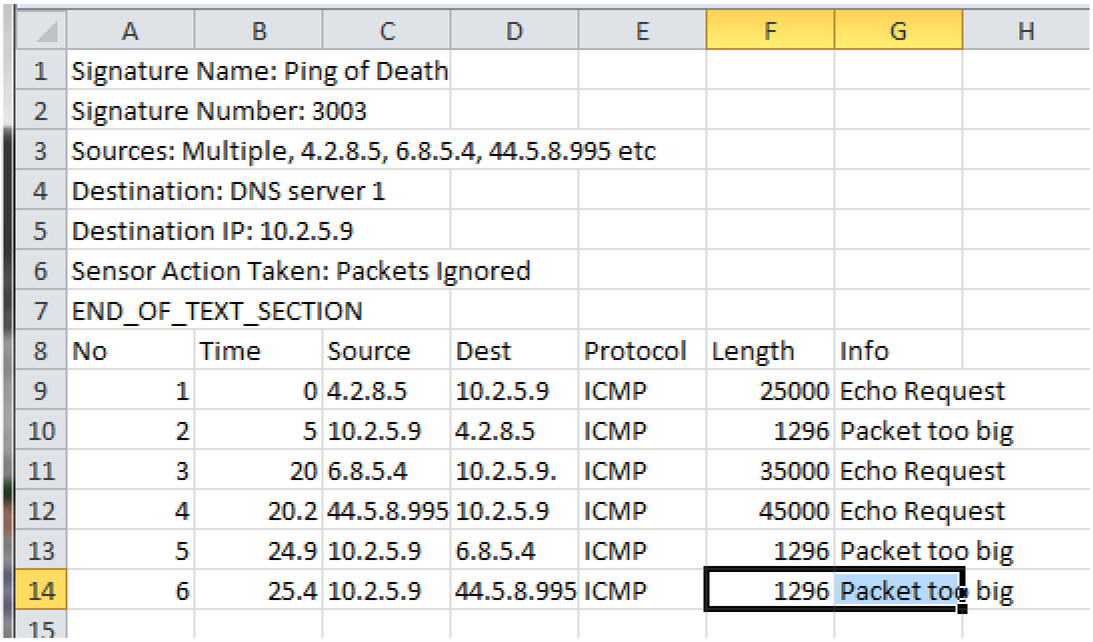
The Colors.csv file indicates how colors are used to represent the levels of alerts. An example of this file is shown in Table 19 below. While the colors currently used match ArcSight, researchers may be interested in studying how different colors affect subjects, so they may change the color scheme.

Table 19: Example Colors.csv File

Number	Color
1	Blue
2	Green
3	Yellow
4	Orange
5	Red

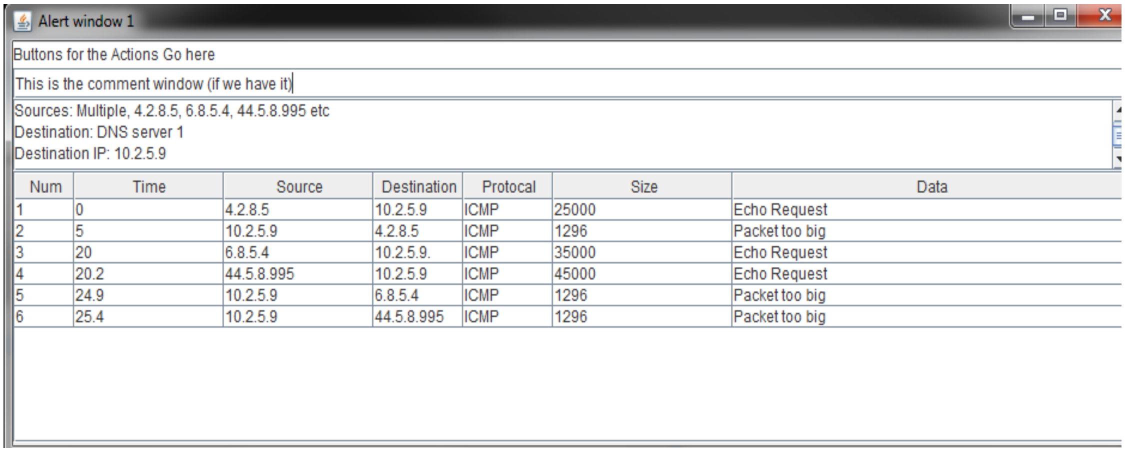
Popup.csv.

This file holds the information that will appear when the subject clicks on the alert. This could include basic information about the alert, such as the action the sensor took, the signature number, along with other relevant information. There is one popup file per alert, but it’s possible that the same alert could be used in multiple trials, so the trails would both use the same Popup File. An example of the csv file is shown in Figure 24 below, with an example of a corresponding Investigation window in Figure 25.



	A	B	C	D	E	F	G	H
1	Signature Name: Ping of Death							
2	Signature Number: 3003							
3	Sources: Multiple, 4.2.8.5, 6.8.5.4, 44.5.8.995 etc							
4	Destination: DNS server 1							
5	Destination IP: 10.2.5.9							
6	Sensor Action Taken: Packets Ignored							
7	END_OF_TEXT_SECTION							
8	No	Time	Source	Dest	Protocol	Length	Info	
9	1	0	4.2.8.5	10.2.5.9	ICMP	25000	Echo Request	
10	2	5	10.2.5.9	4.2.8.5	ICMP	1296	Packet too big	
11	3	20	6.8.5.4	10.2.5.9.	ICMP	35000	Echo Request	
12	4	20.2	44.5.8.995	10.2.5.9	ICMP	45000	Echo Request	
13	5	24.9	10.2.5.9	6.8.5.4	ICMP	1296	Packet too big	
14	6	25.4	10.2.5.9	44.5.8.995	ICMP	1296	Packet too big	
15								

Figure 24: Sample Popup csv file



Buttons for the Actions Go here

This is the comment window (if we have it)

Sources: Multiple, 4.2.8.5, 6.8.5.4, 44.5.8.995 etc
Destination: DNS server 1
Destination IP: 10.2.5.9

Num	Time	Source	Destination	Protocal	Size	Data
1	0	4.2.8.5	10.2.5.9	ICMP	25000	Echo Request
2	5	10.2.5.9	4.2.8.5	ICMP	1296	Packet too big
3	20	6.8.5.4	10.2.5.9.	ICMP	35000	Echo Request
4	20.2	44.5.8.995	10.2.5.9	ICMP	45000	Echo Request
5	24.9	10.2.5.9	6.8.5.4	ICMP	1296	Packet too big
6	25.4	10.2.5.9	44.5.8.995	ICMP	1296	Packet too big

Figure 25: Sample Popup csv Design

Chats.csv.

This file holds the information for when all the chats will play and what they will do. This will include the time they play, the text displayed, and a wav file to play, though the current version of the STE does not implement playing wav files. This could also include how long until the communication disappears if this is implemented. There will be one Chats File per trail. An example of this file is shown in Table 20 below.

Table 20: Sample Chats.csv file

Chat Num	Time	Sender	Text	Wav File	Group
1	3000	Teammate 1	Here is a chat	None	-1[4]
2	6000	Team leader	Here is another chat	None	2

Google.csv.

This file holds all the information that will be displayed when the subject looks up a keyword. These keywords could be signature number, a type of alert (i.e., DDoS, HTTP), or an IP Address, with the corresponding text that will display when the subject searchers it. Each trial needs one file, but the same file is likely able to be used across a large number of trials. An example of this file is shown in Table 21 below.

Table 21: Sample Google.csv File

ID	Keyword	Text
1	22	Signature 22 is triggered by a scan of numerous ports on a network. This is not malicious on its own, but may indicate an attack on that IP is pending
2	HTTP	This web protocol is used to request web pages from a server. This protocol normally runs on port 80.

Output

The output is split into 4 main parts. First, an event file based on all the actions taken. This file includes whenever the subject selected something, opened a window,

[4] -1 represents that it isn't part of a group.

flagged an alert as a threat, etc. The second type of output keeps track of the decisions made for each alert, and what the correct decision was. This allows the researcher to parse the file and, based on the correct/incorrect decisions, allocate a score for the subject. The third output is a record of the mouse and keyboard movements that would allow researcher to replay the trial.

Appendix B: NASA TLX

The NASA TLX is a subjective workload rating scale developed by NASA (Hart, 1988). A subject is asked to circle the tick mark best corresponding to their perceived workload in each of the 6 categories. The TLX form used for this research is shown in Figure 26 below.

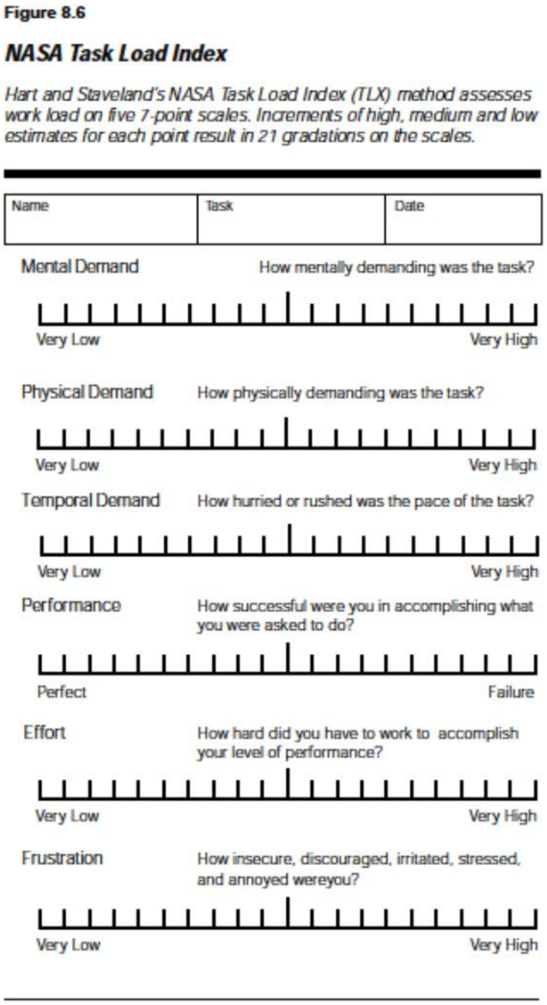


Figure 8.6

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Figure 26: NASA TLX

Appendix C: Post Experiment Interview Questions

1. What processes did you use to decide which alerts to investigate?
2. When investigating the alerts was there a series of steps you normally took?
3. Did you notice seeing the same alert multiple times?
    a. If so, did this make it so you could make a decision faster?
4. Did you ever feel bored by the demands of the task?
5. Did you ever feel rushed by the demands of the task?
6. Were you ever not sure about how to mark an alert, and if so, what did you do?
7. Did you pick up on any patterns in the way the alerts came in, and if so, did you try to adapt your strategy?
8. How much do you feel your cyber knowledge affected your capabilities to perform the task?
9. Did you keep an eye on new alerts coming in as you were investigating one?
10. If not, why not?
11. Did you feel that the trials were too long such that staying focused became a problem?
12. Did you feel you were given enough information to perform the task?
13. How would you rank the difficulty of the 4 conditions?

Appendix D: IMPRINT Design

This Appendix explains the details of the IMPRINT model that was built in section 3.3. As explained in chapter 3.3, the events in IMPRINT are organized in a tree like structure. Each event is either a leaf node, which is a single sub-task, or a function which is composed of several other nodes. This appendix explains each of the functions and the nodes that compose it. In addition, the Appendix provides a table of model wide variables that explains what values they are set at, and their purpose.

IMPRINT Overview.

The IMPRINT model is a series of nodes that track the human's attention throughout the system. In addition, the model may also track automated activities, such as keeping track of the time, or automatically updating variables. The arrows indicated the chain of processes. Any time a node has arrows to multiple nodes, it may take up to all paths. If the node has an M at the end of it, that means it can from zero to all paths, while if it has a T, then it is restricted to exactly one path.

Each low level node may have code snippets in it written in C#. This code can be evaluated before the node occurs, at the start of the node occurring, where a condition must be true for the node to be processed. If the condition is not met, IMPRINT will continue checking until the condition is met. Lastly the code can be run to decide which paths will process next.

For the purposes of this model, purple nodes with rounded corners are low level nodes, not being composed by other nodes. Light grey rectangle nodes are functions that do contain other nodes within them. Lastly, dark grey nodes indicate automatic processes.

These nodes do not indicate tasks performed by the human, but instead are performed by the system. This includes adding new alerts to the system, keeping track of the time, and updating certain variables when needed.

Each node has an identification number, and a name assigned to it. The nodes' identification number does not represent the order in which the nodes are executed.

Page 1: Root

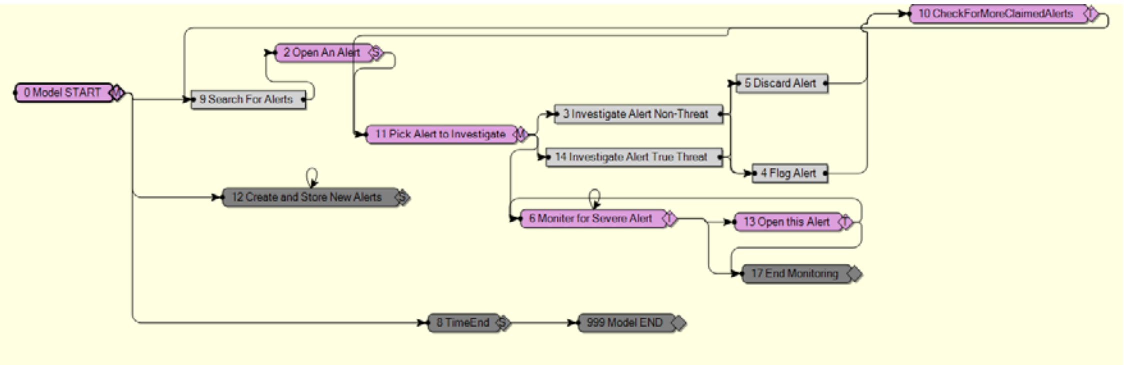

10 CheckForMoreClaimedAlerts
2 Open An Alert
0 Model START
9 Search For Alerts
5 Discard Alert
3 Investigate Alert Non-Threat
11 Pick Alert to Investigate
14 Investigate Alert True Threat
4 Flag Alert
12 Create and Store New Alerts
6 Monitor for Severe Alert
13 Open this Alert
17 End Monitoring
8 TimeEnd
999 Model END

Figure 27: Root Level Diagram

Figure 27 shows the root level diagram of IMPRINT. The root function contains all other functions and nodes in the model. The model starts at node 0, "Model START." This node initializes the variables that determine the condition the model is running, including the Alerts Per Minute (APM) that enter the system, and the distribution of the severity of the alerts. After this, the model splits into 3 paths, Node 9, the Search for Alert function, Node 12, Create and Store New Alerts, and Node 8, TimeEnd. Nodes 8 and 12 are both automatic processes, not reflecting the human's cognitive process. Create and Store New Alerts is an event that occurs at the start of every minute of the trial, and adds new alerts to the system in a number equal to the APM value. TimeEnd counts up to

10 minutes (the length of the trial), and after this time passes, the path continues to Node 999, Model END which kills the model, ending the simulation.

Node 9, “Search for Alerts” is a function that models the process of looking through the alerts in the system to find one to investigate. The details of this node are left to a later section. Once an alert has been identified, the path continues to Node 2, “Open An Alert”, which opens the alert for investigation and also includes code to clean up variables from “Search For Alerts”. Once this node is complete, the path continues to Node 11, “Pick Alert to Investigate”. This node identifies all the alerts that have been open so far and chooses the oldest alert that has been opened to investigate. After this, the path splits, one path to either Node 3 or 14 which both investigate the alert, and a second path to Node 6, “Monitor for Severe Alerts”. “Monitor for Severe Alerts” checks if there are any unclaimed very high alerts in the system. If any of these alerts are found, the path proceeds to Node 13, “Open This Alert” which opens the alert, similar to “Open An Alert”. Once the alert has been opened, this path returns to “Monitor for Severe Alerts” to continue monitoring for alerts. Both “Monitor for Severe Alerts” and “Open This Alert” are connected to node 17, “End Monitoring”. This node serves as a way to get out of the monitoring cycle which only happens while an alert is being investigated. Otherwise there was a possibility that multiple paths would be taken through Nodes 6 and 13 simultaneously.

Nodes 3, “Investigate Alert Non-Threat,” and 14, “Investigate Alert True-Threat” are nearly identical, both serving the purpose of investigating an alert. While the simulated operator doesn’t know if any given alert is a threat or not, this information is accessible to IMPRINT, so the node taken reflects if the alert is a threat or not.

Once the alert has been investigated, the path proceeds to either, Node 4, “Flag Alert”, or Node 5, “Discard Alert”, which are explained in more detail in a later section.

Once the alert has been flagged or discarded, the model then moves to Node 10, “Check For More Claimed Alerts”. This node provides some clean up on the variables used during the investigation. The node also represents the user checking if there are other alerts open to investigate. If there are other alerts that have been opened, then the path proceeds to Node 11 to pick the next alert to investigate, or Node 9, to search for more alerts to open.

Table 22 below indicates the estimated time for all non-function nodes at this level. For all these tables, if the time is a range of values; the distribution is a rectangular distribution with the values being the end points.

Table 22: Times for Nodes in Root Level

Node Number	Node Name	Time	Reason
2	Open An Alert	0.4 seconds	Estimate
6	Monitor for Severe Alert	2 seconds	Estimate
10	Check for More Claimed Alerts	2 seconds	Estimate
11	Pick Alert to Investigate	1 second	Estimate
13	Open This Alert	0.4 seconds	Estimate; made to be same as node 2

Page 2: Search for Alerts

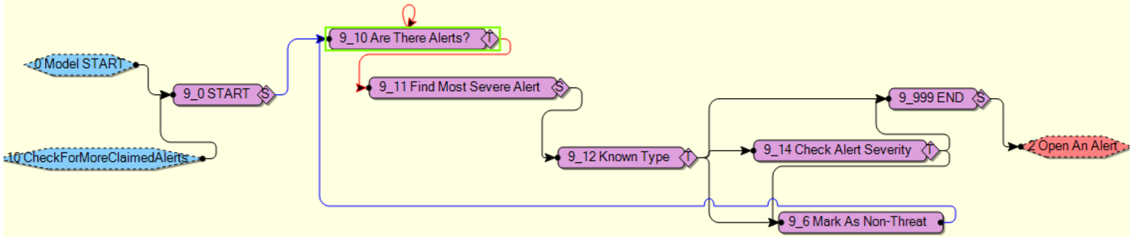


Figure 28: Search for Alerts Function

Figure 28 shows the Search for Alerts function. For functions, there are two additional types of nodes, both represented by hexagons. These nodes indicate connections with nodes outside of the function. Blue hexagons, as seen on the left side of the figure indicate entrances into this function, while red hexagons, as seen on the right side of the screen, represent where the path can go after this function is complete.

Once the path enters this function, the starting node is 9_0, start, which is a placeholder node that doesn't perform any computations. The path then goes to node 9_10, "Are There Alerts". This node checks if there are any unhandled alerts in the system. If there aren't unhandled alerts, the node loops back on itself until there are. If there are alerts in the system, then the path continues to Node 9_11, "Find Most Severe Alert." This node identifies the most severe type of alert in the system, and identifies one of this type of alert to decide if it should be investigated. The path then moves to 9_12, "Known Type." This node checks if the alert in question is of a type where it's known if it should be discarded or investigated. If it is known that the alert should be discarded, then the path moves to Node 9_6, "Mark As Non-Threat" where the alert is discarded, and then the path loops back to "Are There Alerts?". If it is known that based off the type the alert should be investigated, then the path moves onto Node 9_999, "End" where the path will exit the function and then open up the alert. If based off the type, there isn't enough information to decide, then the severity is checked. If the alert is of a high enough severity, then the alert is investigated, otherwise, it is discarded. Table 23 below shows the times for the nodes within Search for Alerts.

Table 23: Time for Nodes in Search for Alerts

Node Number	Node Name	Time	Reason
6	Mark as Non-Threat	1.16 seconds	Based off micromodel with estimated distance to move mouse plus the micromodel motor function to click the mouse
10	Are There Alerts	1 second	Estimate
11	Find Most Severe Alert	1-3 seconds	Estimate
12	Known Type	2 second	Estimate
14	Check Alert Severity	0.5 seconds	Estimate

Page 3: Investigate Alert

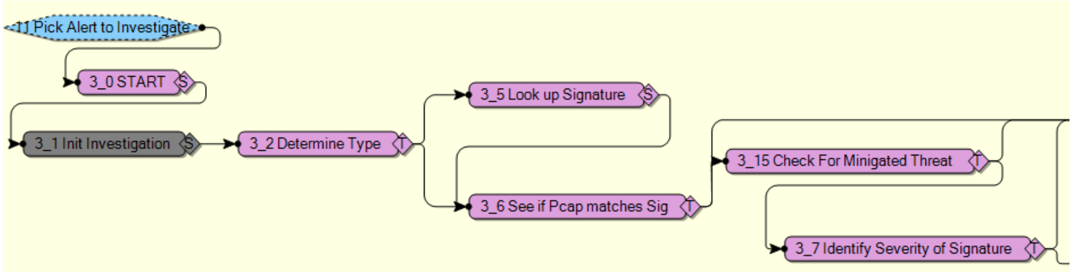

11 Pick Alert to Investigate
3_0 START
S
3_1 Init Investigation
S
3_2 Determine Type
T
3_5 Look up Signature
S
3_6 See if Pcap matches Sig
T
3_15 Check For Minigated Threat
T
3_7 Identify Severity of Signature
T

Figure 29: Investigate Alert Part A

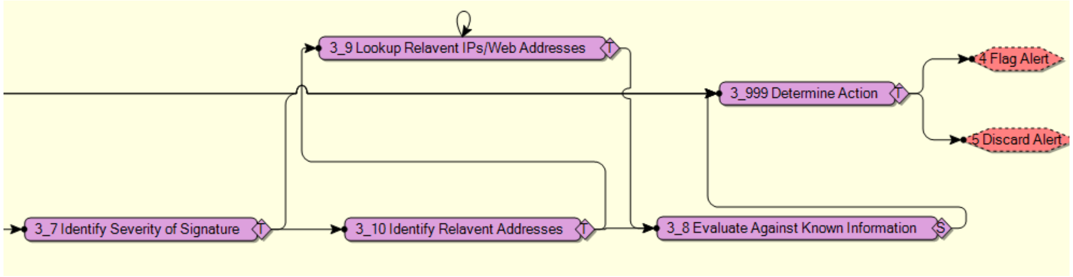

3_9 Lookup Relavent IPs/Web Addresses
T
3_999 Determine Action
T
4 Flag Alert
5 Discard Alert
3_7 Identify Severity of Signature
T
3_10 Identify Relavent Addresses
T
3_8 Evaluate Against Known Information
S

Figure 30: Investigate Alert Part B

Figures 29 and 30 illustrate the Investigate Alert Functions.

The Investigate Alert Functions identifies if an alert represents a true threat or not. Both Investigate Alert functions (Nodes 3, “Investigate Alert Non-Threat” and 14, “Investigate Alert True Threat”) have the same logical paths between them, the only difference being that Alerts that are not a threat are investigated in Node 3, and alerts that are a threat are investigated in Node 14. This split simplifies the code required within the subnodes within the functions, however it does not affect the times or logic required, so any discussion involving, “Investigate Alert Non-Threat” also applies to “Investigate Alert True Threat.”

Nodes 3_0 and 3_1, “Start” and “Init Investigation” initialize the variables, and store the needed information about the alert that’s being investigated. The first human action as part of this function is Node 3_2, “Determine Type.” This node simulates the operator checking the alert for the type it is. After this node, the path branches. If the operator needs to look up the signature, then the path goes to Node 3_5, “Look up Signature,” however, the operator may already know what the signature should look like, so in that case, the path skips ahead to Node 3_6, “See if Pcap Matches Sig.” This node represents the operator looking at the pcap file, and seeing if it matches what the signature said the alert should look like. If the pcap doesn’t match, then the alert is not a threat, and the path advances to Node 3_999, “Determine Action.” From this node, the path exits the function, either going to “Flag Alert” or “Discard Alert.”

If at “See if Pcap Matches Sig”, the pcap does match the signature, then the investigation continues and the path continues to Node 3_15, “Check for Mitigated Threat.” This node checks of the alert was mitigated by a system defense, such as a firewall or system patch. If the alert is mitigated, then it is not a threat, and the path continues to “Determine Action.” Otherwise, the path continues to Node 3_7, “Identify Severity of Signature.” This node checks how severe the action in the alert is. If the alert is either severe enough to definitely need flagged, or the alert is definitely benign activity, then the path continues to “Determine Action.” If there is not enough information to tell either way, then the path continues to Node 3_10, “Identify Relevant Addresses.” This node identifies if there are IP addressed that need to be examined. If there are IP addresses that need checked, the path continues to Node 3_9, “Lookup Relevant IP/Web Addresses” This node checks the IPs identified previously. After the IPs (if any) are checked, the path continues to Node 3_8, “Evaluate Against Known Information.” This node compares the information seen in the alert against any outside information. The path then continues to “Determine Action” which ends the function. Table 24 shows the times for the nodes within the Investigate Alert functions.

Table 24: Times for Investigate Alert Function

Node Number	Node Name	Time	Reason
2	Determine Type	1-5 seconds	Estimate
5	Look up Signature	4-14 seconds	Estimate
6	See if Pcap Matches Sig	5-15 seconds	Estimate
7	Identify Severity of Signature	2-8 seconds	Estimate
8	Evaluate Against Known Information	2-6 seconds	Estimate
9	Lookup Relevant IP Addresses	2-12 seconds	Estimate
10	Identify Relevant Addresses	2-8 seconds	Estimate
15	Check for Mitigated Threat	3 seconds	Estimate

Page 4: Flag Alert

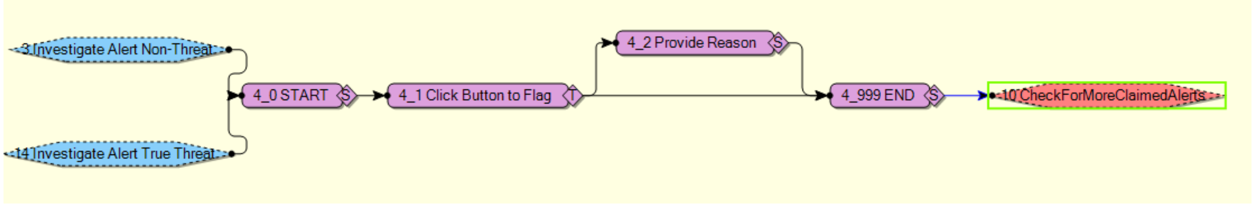

3 Investigate Alert Non-Threat
14 Investigate Alert True Threat
4_0 START
4_1 Click Button to Flag
4_2 Provide Reason
4_999 END
10 CheckForMoreClaimedAlerts

Figure 31: Flag Alert Function

Figure 31 above shows the Flag Alert function.

Function 4, Flag Alert, is the operator flagging an alert as a true threat after the investigation. The path can enter this function from either of the Investigate Alert Functions. Node 4_0, "START" initializes the required data. Node 4_1 "Click Button to Flag" makes the alert as a threat. Depending on the way the system is being implemented, the path can continue to Node 4_2, "Provide Reason" which is where the operator enters the reason for the alert being a threat. Once the reason is provided, or if no reason is needed, then the path continues to the end of the function, Node 4_999. The end node includes code that records that the alert was flagged and modifies the required variables.

Table 25 below shows the estimated times for the nodes within this function.

Table 25: Times for Flag Alert Function

Node Number	Node Name	Time	Reason
1	Click Button to Flag	1.36 seconds	Estimate based off micro model for curser movement and motor function to click.
2	Provide Reason	3-9 seconds	Estimate

Page 5: Discard Alert

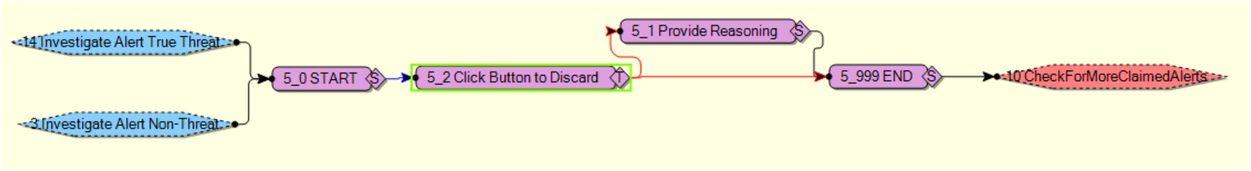

14 Investigate Alert True Threat
3 Investigate Alert Non-Threat
5_0 START
S
5_2 Click Button to Discard
T
5_1 Provide Reasoning
S
5_999 END
S
10 CheckForMoreClaimedAlerts

Figure 32: Discard Alert Function

Figure 32 above shows the Discard Alert function.

Function 5, Discard Alert, is the operator flagging an alert as a non-threat after the investigation. The functionality is identical to Flagging the alert. Node 5_2 marks the alert as a non-threat, Node 5_1 provides the reasoning if the system requires it, and then Node 5_999 ends the function, modifying the required variables.

Table 26: Times for Discard Alert Function

Node Number	Node Name	Time	Reason
1	Click Button to Discard	1.36 seconds	Estimate based off micro model for curser movement and motor function to click.
2	Provide Reason	3-9 seconds	Estimate

Table 27 below provides a listing of all the variables used in the model, along with a short description, their range of values, and the places they are used.

Table 27: Table of IMPRINT Variables

Variable Name	Purpose/Description	Range of Values	Places Used
actionToTake	Stores the decision made when investigating an alert	+1: Threat 0: No Decision -1: Non-threat	Functions 3 and 14
AlertsPerMin	How many alerts enter the system each	1-10, depending on condition	Node 0, to initialize value.

	minute		Node 12, to create new alerts
BoolLookupRequired	Stores if the alert when being investigating requires a signature look up	True/False	"Determine Type" and "Look up Signature" within Investigating alert
BoolSevereAlert	Stores if the alert when being investigating is severe enough to be a true threat	True/False	"Identify Severity of Signature" within Investigating alert
ClaimedAlerts	An array that holds values for alerts current claimed	Values in array range from 1-10 based off the type and severity of alert	
Clock	Time since the trial began		Used by IMPRINT
Condition	The number of the current condition	1-210, depending on the current condition	Node 0 to initialize values
CurrentlyInvestigating	Keeps track if an alert is currently being investigating	True/False	Used in Discard and Flag Alert Functions
InvestigateAlertThreat	Stores if the alert currently being investigated is a threat or not	1=no threat 2=threat	Investigate alert functions Also used in flag/discard functions to check of operator is right
InvestigateAlertType	Stores the type of the alert currently being investigated (Very Low-Very High)	1-5 1= Very Low 2= Low 3= Medium 4= High 5= Very High	Flag/Discard alert functions to aid in cleaning up the categories when an alert has been claimed
Monitor Threat	Used to store if the alert being opened while monitoring for severe alerts is a true threat or not	1= Not a threat 2= Threat	Monitor for Severe Alert, and Open This Alert Nodes
NoThreatDiscard	Stores a count of how many Non-Threats have been discarded	Starts at 0, goes up as alerts are discarded	Used in Discard Alert Function. Also output in the snapshots

NoThreatFlag	Stores a count of how many Non-Threats have been flagged	Starts at 0, goes up as alerts are flagged	Used in Discard Alert Function. Also output in the snapshots
NumAlertsClaimed	Stores a count of how many alerts are currently claimed	Starts at 0, ranges from 0-20 based on the size of the array for Claimed Alerts	Used for Opening Alert Functions, and Discarding/Flag Alert Functions
NumHighThreat1	Stores a count of how many High alerts that are not threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
NumHighThreat2	Stores a count of how many High alerts that are threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
NumLowAlerts1	Stores a count of how many Low alerts that are not threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
NumLowAlerts2	Stores a count of how many Low alerts that are threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
NumMedAlert1	Stores a count of how many Medium alerts that are not threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
NumMedAlert1	Stores a count of how many Medium alerts that are threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
NumThreatsClaimed	Stores a count of how many true threats are currently	Starts at 0, goes up and down throughout the	Output in the snapshots

	claimed	trial	
NumVeryHighThreats1	Stores a count of how many Very High alerts that are not threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes. Also used in Monitor for Severe Alerts
NumVeryHighThreats2	Stores a count of how many Very High alerts that are threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes. Also used in Monitor for Severe Alerts
NumVeryLowThreats1	Stores a count of how many Very Low alerts that are not threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
NumVeryLowThreats2	Stores a count of how many Very Low alerts that are threats are in the system but not yet claimed	Starts at 0, goes up and down throughout the trial	Search for Alerts function, and Open Alert Nodes
PcapMatch	Tracks if the pcap matches the signature when investing an alert	True/False	Investigating alert functions
PercentH	The percent of total alerts that are High	Varies from 10-30 based off condition	Create and Store New Alerts Node
PercentL	The percent of total alerts that are Low	Varies from 15-35 based off condition	Create and Store New Alerts Node
PercentLookupRequired	The percent chance that the subject will need to look up the signature to know what to look for	80, though this is an estimate	Investigate Alert Functions
PercentMed	The percent of total alerts that are Medium	20	Create and Store New Alerts Node
PercentSensorHandled	The percent of alerts that are not threats because the sensor	35, though this is an estimate	Investigate Alert Functions

	handled them.		
PercentSevere	The percent of alerts that are severe enough to be threats	50	Investigate Alert Functions
PercentThreatH	The percent of High alerts that are threats	10	Create and Store New Alerts Node
PercentThreatL	The percent of Low alerts that are threats	1	Create and Store New Alerts Node
PercentThreatM	The percent of Medium alerts that are threats	5	Create and Store New Alerts Node
PercentThreatVH	The percent of Very High alerts that are threats	20	Create and Store New Alerts Node
PercentThreatVL	The percent of Very Low alerts that are threats	0	Create and Store New Alerts Node
PercentVH	The percent of total alerts that are Very High	Varies from 0 to 20 depending on condition	Create and Store New Alerts Node
PercentVL	The percent of total alerts that are Very Low	Varies from 15 to 35 depending on condition	Create and Store New Alerts Node
ScoreAction	A performance score based on actions taken so far. Note that this score was not used in the experiments discussed in this thesis	Starts off at 0, can go into positives or negatives based on decisions made	Search for Alerts, Flag Alerts, and Discard Alert functions
SearchAlertType	The type (Very Low-Very High) of the alert being looked at to decide if it should be investigated	1-5	Search for Alert function, and Open An Alert Node
SearchAlertThreat	If the alert being looked at to investigate is a threat or not	1= not a threat 2= threat	Search for Alert function, and Open An Alert Node
Severity	The Severity Level of the condition, effects the distribution of the Levels of alerts	1-21 depending on condition	Used in the start node to initialize the condition

ThreatDiscard	A count of how many true threats have been discarded	Starts at 0, goes up throughout the trial	Discard Alert Function, also outputted in the snapshot
ThreatFlag	A count of how many true threats have been flagged	Starts at 0, goes up throughout the trial	Discard Alert Function, also outputted in the snapshot

Appendix E: Interview Questions and Responses from SMEs

This appendix provides questions asked during interviews with SMEs along with the answers, in addition to other information obtained through the interviews. Unless otherwise noted, questions and answers preceded with Q and A the exact questions asked and answers provided, while other text provides a summary of information learned.

Interview with Maj. Mike Winn, Army, held April 29 2014

- During his time as an "operator" Maj. Winn was mainly in a management/organizational role, higher up from what this research entails.
- Cyber Operator typically refers to an offensive person, who are heavily overseen from legal and other people who must approve actions
- Information Assurance Analyst refers to the lower level defender, the ones who monitor the network activity.
- Information Assurance Analysts have a very technical understanding of what goes on, but don't always have the big picture fully understood.
- Maj. Winn advices us that we should look for these Information Assurance Analysts for those with experience more closely related to the research.
- The job of Information Assurance is very mundane much of the time.
- Part of their job is to generate reports for those higher up like Maj Winn to understand what's going on.

Interview with Maj. John Rice, Air Force, help May 29 2014

- 90% of interfaces are COTS (commercial off the shelf)
- The 33rd Network Warfare Squadron is the focus of discussion
- Signatures are provided by either commercial entities, or in rare cases, added as a result of government investigations.
- All traffic dumped into ArcSight ESM
- Monitors traffic to Air Force gateway
- The Air Force gateway is the inspection point that looks at everything going in and out.
- Alert information goes to ArcSight
- Task perspective
  - Alerts flagged to be visible show up
  - Certain things alerted with various rating
    - Example, failed logins

- Alerts continuously scroll
- When the operator sees an alert marked high, they click on it to learn more.
- They can get some packet information, or some documentation behind alert to decide if it's real.
- A team of ~20 guys working at once, each one takes a single alert at a time
- When one is viewing an alert, they don't see the others coming in.
- Alerts are missed.
- Each alert takes 1-2 min to decide if it's important, or not.
- If it is, they forward to the lead analyst
- ArcSight can do some correlation to raise an event's severity if it detects certain information

Interview with Gateway Team of 33rd NWS, answered received August 11 2014

This interview was not held in person as the others have been, but was done by emailing a list of questions to a commander at the 33rd NWS who then asked the Gateway Team to answer them. This remainder of this section is a direct copy of the file received from the 33rd NWS.

"Since I'm not sure of your background with ArcSight, I would like to touch on a few things to make sure we're proceeding from a common starting point. Over the years I've noticed some common misperceptions regarding the functionality of ArcSight.

"ArcSight is not an Intrusion Detection/Prevention System. It's a management system that consolidates and normalizes traffic from devices and applications on the network. This allows analysts to see events from multiple devices and ArcSight sites from around the world on one screen.

"ArcSight normalizes traffic from different devices. One device might call an IP 'Sender Address' and one might call it 'Source IP'. ArcSight would normalize this to 'Source Address'.

The Task

Q1) How does ArcSight classify alert levels, and how are these levels displayed? Is it just a number, or color, or something else?

A1) By 'alert level' I believe you mean what we call 'Priority'. 0 thru 10, Very Low to Very High. ArcSight can make adjustments to this rating. ArcSight takes the Severity level assigned from the original device (technically from the SmartConnector that feeds the events from the device into ArcSight) and runs it through a calculation that takes into account the original severity, the model confidence (whether or not the target asset is modeled in ArcSight ESM), relevance (relevance of the event to the asset), severity (has the system been attacked or compromised before), asset criticality (how important the asset has been identified as).

ArcSight priorities run from 0 thru 10. 0-2 = Very Low, 3-4 = Low, 5-6 = Medium, 7-8 = High, 9-10 = Very High. Each group has a color associated with it which helps with visual recognition. The numbers are used for filtering and reporting purposes.

Q2) Does ArcSight change what an alert is classified as based on other circumstances? For example, 5 failed logons might be classified as a level 2 alert most of the time, but if this is accompanied by a suspected DoS attack, this is raised to a level 3 alert.

A2) Based on A1 above, it can. However it depends heavily on the ability to model the network and get the assets listed in ArcSight. Also, some devices might only use a 1 thru 5 scale. ArcSight would translate these into the 0 thru 10 scale that ArcSight uses.

Q3) What things affect the severity of the alert?

A3) See A1.

Q4) What information does ArcSight display about an alert as it comes in? For example, source, type, description?

A4) ArcSight has hundreds of fields that can be populated with information. Some relate to the events, some relate to the devices that process the events. Some are customizable. Admin and content developers can even create some fields using local variables to define the data that goes into the field. Field Sets (a particular combination of fields) can be saved and attached to the channels that analysts work out of, allowing for a standard default view. Individual analysts can easily add or remove fields.

A default Field Set may contain Time, IP's, Ports, Event Name, and Priority fields. In many cases our default Field Sets contain enough information to allow analysts to close many False Positives without deeper analysis. If needed, he operator can view the event in the Inspect/Edit window, which shows all of the ArcSight fields.

Q5) What information in this alert makes an analyst decide to click on it or not?

A5) Pretty much all events are 'clickable'. The goal is to either close all events as non-malicious, reportable in some way, or in need of further review. A combination of the Event Name, IP's, and Priority will help the analyst decide which events to work first and how deep they have to go in their analysis in order to determine if the event is malicious or not. Generally, analysts work from oldest to newest.

However, they keep an eye out for higher priority events or events that have a higher likelihood of being malicious. By adjusting their Field Sets the analyst may be able to see enough information on their screen to be able to close an event as non-malicious without having to do deeper research. If needed, the analyst can retrieve the event from the full packet capture device and review it in Wireshark.

Q6) Can you open more than one alert at a time?

A6) Newer versions of ArcSight allow an analyst to have more than one event inspector window open at the same time, if they choose.

Q7) Is there anything beyond the particular alert that influences your decision about whether or not to click it?

A7) Historical experience helps analysts decide if they need to review an event beyond the data that is available on their screens. Local knowledge from briefings and pass-on logs may alert analysts to be aware of certain specific activity to look for or that a particular base is being targeted.

Q8) How long does an analyst have to decide if they want to click on the alert or not?

A8) As stated above, all events are supposed to be worked, that is closed as non-malicious or forwarded for further review.

Q9) What happens to an alert when you click on it?

A9) Clicking/Inspecting an event opens it in ArcSight's Inspect/Edit window. This allows the analyst to see all of the information that ArcSight has on the event.

Q10) What happens to alerts you don't click on?

A10) The analysts work out of channels. These channels use filters to limit the events to only that which we have deemed as ‘actionable’, that is events that could not be categorized as False Positive and require analyst review. The channels have a time frame, maybe one hour long. When events have not been worked and closed before the time allotted for that particular channel, they ‘fall off’ the bottom. The events remain in ArcSight, they just no longer match the filter criteria for that channel. Time permitting, senior analysts have access to channels with longer time frames and try to catch those missed by the primary analysts.

Q11) Once an alert is brought up, what information is then displayed?

A11) See Q9.

Q12) From here, how does an analyst decide if it’s important or not?

A12) By reviewing the event name, reviewing the IDS/IPS signature to determine what caused the event to fire, and pulling up the event in Wireshark, the analysts determines if the events is non-malicious or not. If it is non-malicious, the analyst annotates the event and marks it as closed. If the analyst can’t determine that the event is non-malicious, he or she begins a reporting process that forwards the information to the Incident Response Team for further analysis.

Q13) What actions do you take in order to “close” an alert?

A13) Determine the event is non-malicious, enter a note on the event stating what research you did and why you came to your decision, change the event stage to ‘closed’.

Q14) Do you get feedback on the correctness of your decisions?

A14) Lead Analysts periodically review events.

Q15) Is there an estimated number of false positives or false negatives committed at this level?

A15) This is more difficult to determine than it appears. Many of the commercial signatures are ‘loose’ on purpose, so as not to accidently let False Negatives through. We have a process in place to try and identify commercial signatures that are no longer a threat on our network (maybe a patch has been available for many years or the vulnerable asset is no longer on our networks). If the signature can be turned off or set to block we can do that. We can also use some of ArcSight’s filtering and rules capabilities to limit the False Positives that appear on the analysts’ screens.

Q16) How long does it take an analyst to decide if the alert should be sent onward? I’ve heard 1-2 minutes, is this accurate?

A16) We don’t have a hard and fast time frame due to the different research steps that may need to be taken. But once an analyst selects an event they can probably determine if it needs to be forwarded for further review within five minutes.

Team Work

Q17) How many floor analysts are normally working in the same time/place?

A17) Average 3 – 12, depending on break days (weekends) and shift (Days, Swings, Mids).

Q18) Is there communication between them, either in person or electronically via a text interface or audio channel?

A18) We sit in three rows of four. Easy to talk between analysts.

Q19) Do you wear headsets? If so, how do you use them?

A19) We don’t use headsets. Nothing stopping a system from being used, however.

Q20) Could there be some kind of alert broadcast that will affect the way the analysts classify individual alerts? For example, something has happened which would make any lower level alerts with a source of country X need investigating, even if they’re a low enough level to normally ignore.

A20) Not as a formal application. Primarily pass-on logs and shift change briefings. Lead Analysts would pass the word during a shift.

Q21) If one analyst clicks on an alert to investigate it, does it disappear from the screens of the others, or is grayed out, or something similar?

A21) No. When an analyst begins to work an event, they put their name on it. All ops have that field in their Field Set and they would see that someone is already working that event.

Q22) Do analysts discuss alerts while they are working on them?

A22) Yes, as they feel the need.

Q23) What other effects might team members have on an analyst’s job?

A23) Analysts pass on tips for reviewing events. What a True Positive or False Positive looks like. Advice on how to manipulate ArcSight or other devices in order to improve their review procedures.”

Interview with Griffin Team of 33rd NWS

This interview was conducted in the same way as the one with the Gateway team, with the team being provided with a list of questions and then answering them. The

Gateway team monitors the NIPR network traffic, while the Griffin team monitors SIPR network traffic.

In addition, James Hannan, Director of Operations at the 33rd NWS said,

“The tool [ArcSight] is the same on both networks but used quite differently because of internet access on NIPR and the lack of internet access on SIPR. There is significantly more traffic on NIPR and vendor signatures tend to work fairly well. However, on SIPRNet the 33 NWS spends a lot more time customizing signatures to the environment.

“A huge challenge on both networks is signature tuning, most of our signatures have a false positive rate in the 99.999% range. Because of the challenge in tuning the signatures, my operators spend a fair amount of time building custom filters in ArcSight to reduce some of the noise.” (Hannan, 2014)

The remainder of this section is copied directly from the document provided by the 33rd.

“The Task

Q. How does ArcSight classify alert levels, and how are these levels displayed? Is it just a number, or color, or something else?

A. ArcSight classifies alert levels into very low, low, medium, high and very high event severities.

A. The event severities are distinguished from one another by color (default colors can be changed in preferences).

A. The classification levels are parsed into a legible format from various appliances (McAfee, Cisco, etc) as depicted above.

Q. Does ArcSight change what an alert is classified as based on other circumstances? For example, 5 failed logons might be classified as a level 2 alert most of the time, but if this is accompanied by a suspected DoS attack, this is raised to a level 3 alert.

A. No, an alert classification is based upon the signature.

A. This can be modified at the signature level

Q. What things affect the severity of the alert?

A. The severity of the alert is set by the vendor and/or the individual writing the signature.

A. Type of alert affects the severity as well (virus, worm, etc)]

Q. What information does ArcSight display about an alert as it comes in? For example, source, type, description?

A. Date and time of connection, source, target, source port, target port, device host name are but a few Field Sets that can be viewed upon selection within the Viewer panel.

A. Depending upon the data collected from the event, there can be over 50 different possible fields to select from.

Q. What information in this alert makes an analyst decide to click on it or not?

A. All alerts that fire within the application (ArcSight) have to be viewed in order for the alert to be archived.

A. The type of traffic or alert and the severity of the event might affect the analyst’s decision when selecting

Q. Can you open more than one alert at a time?

A. Yes, you can open more than one alert at a time.

Q. Is there anything beyond the particular alert that influences your decision about whether or not to click it?

A. Similar alerts showing (i.e., same source or destination) would affect our decision to view (Scan, etc).

Q. How long does an analyst have to decide if they want to click on the alert or not?

A. Length of time is ultimately a result of the amount of traffic flow within a channel. Everything is dependent on the Time to Live (TTL) of the channel and filter configuration before alerts will clear.

Q. What happens to an alert when you click on it?

A. All Alerts are viewed in the viewer window. Individual alerts selected are displayed in the Inspect/Edit tab.

Q. What happens to alerts you don't click on?

A. All alerts will remain in the channel until the TTL expires.

A. Depending on the data collected by the signature, 128 bytes, will not be able to be viewed unless the alert was selected within a reasonable time (Archived Event)

Q. Once an alert is brought up, what information is then displayed?

A. Everything parsed from the connector, on the appliance, will be viewable in the Inspect/Edit window.

Q. From here, how does an analyst decide if it's important or not?

A. The severity of the signature, and frequency of alert.

Q. What actions do you take in order to “close” an alert?

A. In order to close an alert, you need to annotate the alert as closed.

Q. Do you get feedback on the correctness of your decisions?

A. Feedback is directly related to validation opened, or team members from discussion sessions of particular events in question.

Q. Is there an estimated number of a false positive or false negatives committed at this level?

A. Utilizing the tools in place, for our particular mission, our process is to validate all traffic based on signature sets.

Q. How long does it take an analyst to decide if the alert should be sent onward? I’ve heard 1-2 minutes, is this accurate?

A. Every alert requires different levels of research and/or actions taken.

Team Work

Q. How many floor analysts are normally working in the same time/place?

A. There are usually between two and four analysts within our environment.

Q. Is there communication between them, either in person or electronically via a text interface or audio channel?

A. Communication is vital in Cyber Defense.

Q. Do you wear headsets? If so, how do you use them?

A. Headsets are not needed in our environment.

Q. Could there be some kind of alert broadcast that will affect the way the analysts classify individual alerts? For example, something has happened which would make any lower level alerts with a source of country X need investigating, even if they're a low enough level to normally ignore.

A. Depending on the environment a broadcast could be utilized, but the logic behind Real-Time alerts, worms, virus, and zero-days do not dictate the need for that type of affect.

Q. If one analyst clicks on an alert to investigate it, does it disappear from the screens of the others, or is grayed out, or something similar?

A. When an event is selected, the annotation field populates with the individuals name who is working the event.

A. An alert only disappears from the screen once placed in a closed state (filters dictate this type of action).

Q. Do analysts discuss alerts while they are working on them?

A. Continuously, it is always good practice to have a second set of eyes when working events in question.

Q. What other effects might team members have on an analyst's job?

A. Continuing education/training.

Follow up

Q. Is there anyone else you think we should interview?

A. I would recommend interviewing the commercial sector to get an idea of the way analysis is performed.

A. The commercial sector is not limited to the amount of tools(i.e., messanger, free BSD), updates, and equipment."

Notes from Interview with George Lovell of the 33rd NWS, July 31, 2014

ArcSight (an HP product) compiles reports from a number of different sensors, including but not limited to mail servers, DNS servers, IDS systems. HP has a PDF available for the product if we need help

Each of these sensors has a number of levels for the alerts, which ArcSight displays.

There are 5 levels/colors of alerts:

- Blue – Very Low (ignore)
- Green – Low
- Yellow – Moderate
- Orange – High
- Red - Critical

Analysts normally will look at medium/yellow or higher.

ArcSight information contained in the alert row at the top level includes: Protocol, IPs, Ports, alert Name, Message, Action Sensor took (ie, deleting spam email)

When the analyst clicks on the alert, they get a more information in the inspect/edit panel – which includes all data forwarded by the sender for that event along with information including a packet capture (PCAP) report (similar to WireShark), session information, and other information from the sensor that generated the alert.

It is possible for ArcSight to combine several events together in the correlated event. ArcSight may also display something at a higher level, if it matches a set of site-specific rules[5], this is a correlated event. Correlated events are marked in the (left hand column?) with a lightning bolt.

[5] Analysts or the local site can set up additional rules which increase/decrease the alert level of a type of alert.

There are approximately 6 analysts on a shift. Each analyst reviews about 1k alerts per 8 hour shift. The analysts are using dual 21” monitors – ArcSight is on one screen.

The analyst looking at an alert takes from 2-3 min to up to an hour to decide if the alert should lead to investigation. The process often requires opening up an event and looking at additional info such as packet capture (PCAP) sessions.

Lead analysts usually have more experience. They interact heavily with line analysts, and the interactions are often skill-set dependent.

If the Lead Analyst sees a common false positive, they may include a rule to filter it out; this rule can be temporary or permanent. Often the determination to filter something out is made in conjunction with things the other analysts are seeing or have already handled.

ORDER OF EVENTS FOR HANDLING EACH ALERT (OR GROUP):

1. ArcSight creates an alert which shows up at the (top?) of the screen on a new row
2. The analyst[6] decides to click on it or not.
3. The analyst is supposed to mark the alerts that they are working on (or plan to work on) by checking the annotation column for all the alerts they are working. Analysts may work more than one alert simultaneously (e.g. if they notice a pattern of related alerts). The analyst ID appears in the annotation column for alerts they are working[7]
4. The analyst investigates the alert
    a. If a line analyst determines the event needs to be looked at further, the line analyst gets permission from the lead analyst to create a report of the basic facts (on SharePoint?)
    b. If the line analyst determines there is no action required, he closes the alert[8].
5. The incident response team (IRT) investigates it.

INTERNAL COMMUNICATION: There is a lot of teamwork involved, if someone sees something they don't understand, they may ask another analyst who is an expert in that field. Most of the teamwork is ad-hoc. The analysts communicate directly with each other verbally. An analyst will often will ask another analyst to 'come over and look at this' for items which they want another (or an expert) opinion. Mr. Lovell pointed out that none of this ad-hoc communication gets recorded. (AFIT is assuming there are no mics or headsets, and the analysts don't generally use chat software).

[6] Here, "analyst" can refer to line analyst or lead analyst. All members on a shift are looking at alerts

[7] One issue that Mr. Lovell described is that analysts don't always follow the same sequence when deciding whether to mark an alert or not. Sometimes they browse for a bit before deciding to mark one. Also, they might mark alerts even if they don't end up working them. This ad-hoc marking protocol could lead to race conditions and missed alerts. Mr. Lovell did mention they are trying to be more rigorous about their marking procedures. The new procedure is to tag it ahead of time, saying that you are reviewing it. But this is still a personal process to remember to mark the item as soon as you open it.

[8] Most alerts get "dropped" after initial inspection. We did not ask Mr. Lovell about how they are marked once dropped – is there something that tells other analysts not to look at them?

EXTERNAL COMMUNICATION: Analysts may also get calls from another base if their analysts have a question. Gunter AFB is the hot backup / continuity of operations (COOP) site for 33NWS. They have analysts that are performing the same mission on the same data simultaneously.

FEEDBACK: Analysts normally told if they were right or wrong about a report, but not necessarily quickly. Feedback arrives within a few hours to a few months later. There is a heavier emphasis on negative feedback (you got this wrong)

Follow up Email Questions and Answers with George Lovell

Questions were asked and answered over a period of time from September 2014 to March 2015.

About 1% of alerts that come into ArcSight are sent to the IRT for further investigation.

About 50% of alerts are either very low or low. These are not normally investigated unless there is additional evidence to suggest that they may be important. This could be ArcSight combining several low events together into a higher meta event, or the low events contain a time or IP address known to be involved in a high priority event.

About 20% of alerts are medium.

About 20% of alerts are high.

About 90% of the medium and high alerts are investigated

About 10% of alerts are very high, and 99% of these alerts are investigated.

Whenever an event is marked as non-malicious, this is commented on the alert itself so other analysts can see it in ArcSight, with the degree of detail varying depending on the analyst and the detail required for the alert.

Typically alerts are prioritized based on their priority, but “unique or potentially malicious alerts” are given higher priority despite ArcSight not making them as such. This includes alerts that correlate to known events such as the Heartbleed vulnerability.

“Q: How does an analyst determine if an alert is malicious or benign?

A: “The analyst will need to resolve the ownership of the IPs involved in the connection. This requires using "NSLOOKUP" to get the fully qualified domain name (FQDN) and performing a "whois" lookup using various web sites from metre.net to visiting the various regional internet registries (i.e. APNIC, ARIN, RIPE NCC, LACNIC, AFRINIC, etc.)

“The analyst should review the reputation of referenced web sites in the pcap on McAfee's Site Advisor, Norton's Safe Web, VirusTotal, etc. Also, the analyst should look at the signature page for the IDS/IPS reporting the activity. This should give what the signature is looking for and what the signature hit on. The analyst often finds that the alert/event was triggered by a partial match. Sometimes there will be a full signature match but the context is wrong. Again this comes from reading the signature page to see what is being looked for.

“Reading through the pcap file in Wireshark, the analyst looks for things like RST packets before FIN packets. A session that ends with a RST packet but not a FIN packet results in an incomplete session that the destination computer drops all data from its memory.

“And when all of this research does not provide a definitive answer as to whether the reported activity is malicious or benign, the analyst must rely on experience. We often annotate these events with "Nothing malicious found. Will continue to monitor." Because that is the best we can do.”

Also, the CISCO 2000 series notes, found in the appendix of (Baumrucker, et al., 2003) gives a good example of the type of signatures the 33rd uses, though these exact signatures are likely outdated.

“Q: What general rules does the analyst use to determine if an alert is a true threat?

A: “Because humans write the signatures and heuristics algorithms, we deal with a lot of false positives or more benign than malicious true positives. To determine this, the analyst must look at the packet capture (pcap) of the session. This is the only valid way to determine if the reported event is malicious or not. The analyst/operator looks at several hundred pcaps a day.”

Q: Do you have any estimates for how often alerts are deemed not threats for various reasons.

A: “Approximately, 75% of the false positives are from partial signature hits. The rest are either benign or otherwise defended against by other infrastructure devices.”

Notes from Interview with Maj. Samuel Stone, USAF

Maj. Stone worked at the 33rd as a Crew Commander in 2006, so some of his information may be dated.

Crew commanders typically lead the line analysts, and made the decisions, however they tended to not delve into the technical aspects the analysts did.

The information displayed in the alerts normally included information such as Alert Name, Source IP, Destination IP, Description, with the malicious payload sometimes seen in the description.

The analyst chooses to investigate an alert based on context, such as recent threats being higher priority. Situational Awareness is required to remember these big threats. There is a shift change brief held, where the new shift gets a summary of important information from the various teams. This includes information such as where attacks are coming from, things the IRT saw, etc. The line analysts may also inform the incoming line analysts of what things they were looking at, a sheet of notes.

The alerts take somewhere between 10 seconds to 1-2 minutes to investigate, with 5 minutes being an upper bound. The alerts that can be discarded right away if it’s seen that the system blocked the attack, or was patched, or something like this. False matches are also quite common and can also be quickly discarded. These matches could be a hex string appearing elsewhere in the traffic where it wasn’t a threat.

The alerts could be sent to different teams, for example the IRT or the virus team.

An analyst could request a block from a malicious IP if enough traffic was seen. This would have to be approved higher up, but does start with the Line Analyst.

It is better to flag something as a threat that isn’t than to let a true threat go undetected. False positives are better than false negatives.

Hard alerts are those that take longer to solve, that can’t be dismissed or confirmed with a quick glance.

The possible threat of the alert could be indicated by the time it occurred: A big data transfer during normal working hours vs. one at midnight.

The idea of having one window for information lookup seems reasonable for Maj. Stone. Using a Google search is rare at the line analyst level. Signature documentations the analysts may have access too.

Experienced analysts may be able to remember IP addresses, able to map a certain address to a certain country without needing to look it up.

Bibliography

Army Research Laboratory. (2010, September 1). *Improved Performance Research Integration Tool.* Retrieved from United States Army Research Laboritory: http://www.arl.army.mil/www/default.cfm?page=445

Baumrucker, C. T., Burton, J., Dentler, S., Dubrawsky, I., Osipov, V., & Sweeney, M. (2003). *CISCO Guide to Secure Intrusion Detection Systems.* Syngress Publishing.

Brehmer, B., & Dörner, D. (1993). Experiments with Computer-Simulated Microworlds: Escaping Both the Narrow Straits of the Laboratory and the Deep Blue Sea of the Field Study. *Computers in Human Behavior*, 171-184.

Budiu, R. (2013). *Act-R.* Retrieved from Carnegie Mellon University: http://act-r.psy.cmu.edu/about/

Caldwell, J., & Ramspott, S. (1998). Effects of task duration on sensitivity to sleep deprivation using the multi-attribute task battery. *Behavior Research Methods, Instruments, & Computers*, 651-660.

Champion, M., Rajivan, P., Cooke, N., & Janwala, S. (2012). Team-Based Cyber Defense Analysis. *2012 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, 218-221.

Comstock, J., & Arnegard, R. (1992). *The Multi-Attribute Task Battery for Human Operator Workload and Strategic Behavior Research.* NASA.

Cooke, N., & Shope, S. (2004). Designing a Synthetic Task Environment. *Scaled Worlds: Development, Validation, and Application*, 263-278.

Cooke, N., & Shope, S. (2005). Synthetic Task Environments for Teams: CERTT’s UAV-STE. *Handbook on Human Factors.*

Crandall, B., Klein, G., & Hoffman, R. (2006). *Working Minds, A Practitioner's Guide to Cognitive Task Analysis.* Massachusetts Institute of Technology.

D'Amico, A., & Whitley, K. (2007). The Real Work of Computer Network Defense Analysts. *Proceedings on the Workshop on Visualization for Computer Security*, 19-37.

Fink, G., North, C., Endert, A., & Rose, S. (2009). Visualizing Cyber Security: Usable Workspaces. *Visualization for Cyber Security*.

Finomore, V., Sitz, A., Blair, E., Rahill, K., Champion, M., Funke, G., . . . Knott, B. (2013). Effects of Cyber Disruption in a Distrubuted Team Decision Making Task. *Proceedings of the Human Factors and Ergonimics Society 57th Annual Meeting*, 394-398.

Galster, S. M., & Bolia, R. S. (2005). Decision quality and mission effectiveness in a simulated command and control environment. *Human Performance, Situational Awareness and Automation*, 264-268.

Galster, S., & Bolia, R. (2004). Decision quality and mission effectiveness in a simulated command and control environment. *Human performance, situation awareness, and automation: Current research and trends*, 264-268.

Giacobe, N. (2013). A Picture is Worth a Thousand Alerts. *Publication: Proceedings of the Human Factors and Ergonomics Society*, *57th Annual Meeting*, 172-176.

Guznov, S., Matthews, G., Funke, G., & Dukes, A. (2011). Use of the RoboFlag synthetic task enviorment to investigate workload and stress responses in UAV operation. *Behavior Research Methods* , 771-780.

Hannan, J. (2014, July). (G. Dye, Interviewer)

Hart, S. G. (1988). "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research." . *Advances in psychology 52* , 139-183.

Hellar, B. D., & McNeese, M. (2010). NeoCITIES: A Simulated Command and Control Task Envionment for Experimental Research. *PROCEEDINGS of the HUMAN FACTORS and ERGONOMICS SOCIETY 54th ANNUAL MEETING – 2010`*, 1027-1031.

Hewlett-Packard Development Company. (2012). *Securing your IT infrastructure with SOC/NOC collaboration.*

Liu, P., Erbacher, R., Glodek, W., Etoty, R. E., & Yen, J. (2013). *Human Subject Research Protocol: Computer-Aided Human Centric Cyber Situation Awareness: Understanding Cognitive Processes of Cyber Analysts.* ARMY RESEARCH LAB ADELPHI MD COMPUTATIONAL AND INFORMATION SCIENCES DIRECTORATE.

Lovell, G. (2014, July 31). (G. Dye, & B. Borghetti, Interviewers)

Mancuso, V., Minotra, D., Giacobe, N., McNeese, M., & Tyworth, M. (2012). idsNETS: An Experimental Platform to Study Situation Awareness for Intrusion Detection Analysts. *CogSIMA 2012*, 73-78.

Miller, W. (2010). *The US Air-Force-Developed Adaption of the Multi-Attribute Task Battery for the Assessment of Human Operator Workload and Strategic Behavior*. Air Force Research Labs.

Mitchell, D. K. (2000). *Mental Workload and ARL Workload Modeling Tools.* ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD.

National Initiative for Cybersecurity Education . (2011). *National Cybersecurity Workforce Framework.* Retrieved from http://csrc.nist.gov/nice/framework/

Powers, D. M. (2011). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation.

Rajivan, P. (2011). *CyberCog, A Synthetic Task Environment for Measuring Cyber Situation Awareness.*

Rice, J. (2014, May 29). (G. Dye, C. Rusnock, J. Boubin, & B. Borghetti, Interviewers)

Sohail, A. (2014). Providing Information Security Using ArcSight SIEM in an Organization. *International Journal Of Engineering And Computer Science*, 5364-5368.

Stone, S. (2015, 3 3). (G. Dye, & B. Borghetti, Interviewers)

Voorhees, J. (2007). *Distilling Data in a SIM: A Strategy for the Analysis of Events in the ArcSight ESM.* SANS Institute.

Wilson, G., & Russell, C. (2003). Real-Time Assessment of Mental Workload Using Psychophysiological Measures and Artificial Neural Networks. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 635-643.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* 18-06-2015	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* 20 Aug 2013 – 18 Jun 2016

TITLE AND SUBTITLE

Using IMPRINT to Guide Experimental Design with Simulated Task Environments

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Dye, Gregory W, Civ, USAF

5d. PROJECT NUMBER
15G129

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-15-J-052

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Labs, RHCPA
Dr. Scott Galster, Chief
BLDG 840, Room W200 2510 Fifth Street, WPAFB, OH 45433
937-938-3572 and Scott.Galster@us.af.mil
ATTN: Scott Galster

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RHCPA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Experimental Designs involving Simulated Task Environments aim to explore interesting conditions with human subjects. By using activity simulators such as IMPRINT, it may be possible to identify these conditions of interest without the need for human subjects. This thesis presents research that aims to demonstrate that IMPRINT can be used to predict human performance in a task environment representing the task performed by Network Analysts of the 33rd Network Warfare Squadron. The research is done by examining the task performed by the Network Analysts, and then designing a Simulated Task Environment modeled on this task. A model of the task performed is also built in IMPRINT. With a first iteration, it was found that the IMPRINT model was not able to predict performance in a majority of cases, however the methodology illustrates a starting point that others may use.

15. SUBJECT TERMS
Cyber, STE, IMPRINT

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Brett Borghetti, AFIT/ENG
U	U	U	UU	151	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4612 (brett.borghetti@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18